The two clocks and the innovation window: When and how generative models learn rules

Binxu Wang * **Emma Lucia Byrnes Finn** **Bingbin Liu**
binxu_wang@fas.harvard.edu efinn@college.harvard.edu bliu@g.harvard.edu
Kempner Institute at Harvard University, Boston, MA, USA

Abstract

Generative models trained on finite data face a fundamental tension: their score-matching or next-token objective converges to the empirical training distribution rather than the population distribution we seek to learn. Using rule-valid synthetic tasks, we trace this tension across two training timescales: τ_{rule}, the step at which generations first become rule-valid, and τ_{mem}, the step at which models begin reproducing training samples. Focusing on parity and extending to other binary rules and combinatorial puzzles, we characterize how these two clocks τ_{rule}, τ_{mem} depend on key aspects of the learning setup. Specifically, we show that τ_{rule} increases with rule complexity and decreases with model capacity, while τ_{mem} is approximately invariant to the rule and scales nearly linearly with dataset size N. We define the *innovation window* as the interval $[\tau_{\text{rule}}, \tau_{\text{mem}}]$. This window widens with increasing N and narrows with rule complexity, and may vanish entirely when $\tau_{\text{rule}} \geq \tau_{\text{mem}}$. The same two-clock structure arises in both diffusion (DiT) and autoregressive (GPT) models, with architecture-dependent offsets. Dissecting the learned score of DiT models reveals a corresponding evolution of the optimization landscapes, where attractors emerge at both timescales: rule-valid samples' basins expand substantially around τ_{rule}, while training samples' basins begin to dominate around τ_{mem}. Together, these results yield a unified and predictive account of when and how generative models exhibit genuine innovation.[2]

1 Introduction

Modern generative models learn from finite data, yet we want them to produce novel samples rather than reproduce their training examples. For diffusion models, prior work has examined how data statistics and denoiser inductive biases shape this generalization [Kadkhodaie et al., 2024, Niedoba et al., 2024, Kamb and Ganguli, 2024, Wang and Vastola, 2024, Li et al., 2024b, Lukoianov et al., 2026]. Avoiding copies, however, does not by itself show that a model has learned the data distribution: novel samples may still violate its underlying rules. Such violations are harder to identify in natural data, but synthetic rule-based tasks make them measurable and reveal that, indeed, diffusion models struggle to learn some rules precisely [Wang et al., 2024, Han et al., 2025]. This raises the question of which rules generative models can learn, and under what condition.

Training dynamics offer another perspective into rule learning. Bonnaire et al. [2025] identify an early timescale for improved sample quality and a later memorization onset in diffusion models, leaving a window of high-quality, novel generation. In a hierarchical grammar, Favero et al. [2025] track when rules at different layers are learned and how dataset size affects the balance between valid generation and memorization. Building on this picture, we ask how rule complexity, dataset size,

*Corresponding to binxu_wang@fas.harvard.edu.
[2]Code available at https://github.com/Animadversio/GenRuleLearningDynamics.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

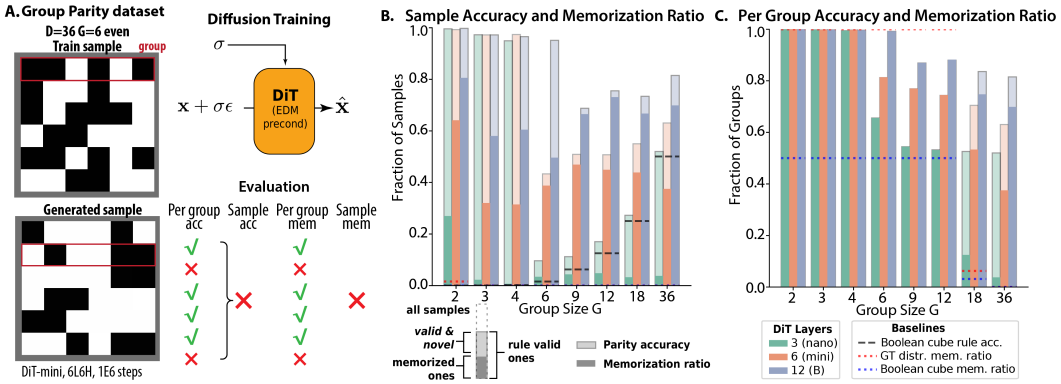


Figure 1: **Rule learning and memorization for group-parity. A.** Structure of the Group Parity dataset and evaluation setup. Each D-dim binary image is divided into groups of size G (here $D = 36$, $G = 6$), with each *group* satisfying even parity. We train DiT models on datasets of $N = 4096$ samples and evaluate model generations in term of accuracy and memorization ratio, at both (**B**) sample and (**C**) group level, across different model sizes. In **B**-**C**, pale bars denote rule accuracy, and dark bars denote the portion of generations that are seen during training (i.e., memorized and valid). The black dashed lines mark the random baseline accuracy, while blue and red dashed lines mark the baseline memorization ratios for the Boolean cube and ground-truth distribution.

model capacity, and data representation affect the timing of rule learning and memorization, and characterize changes in the model around these transitions.

To answer these questions precisely, we use group-level parity, a classic challenge in learning theory [Kearns, 1998, Barak et al., 2022]. Each binary image is divided into groups of G bits, each constrained to have even parity. Because the constraint depends on all G bits in a group, varying G lets us probe rule learning at different interaction orders. The known rule-valid support and training set also provide exact chance baselines for validity and memorization. We track the onset of rule-valid generation, τ_{rule}, and the onset of excess training-set matches, τ_{mem}. Their relative timing determines the *innovation window*, during which the model can generate samples that are both valid and novel.

We first ask what determines the two onsets (Sec. 3). In group parity, increasing rule complexity G sharply delays τ_{rule}, whereas increasing dataset size N delays τ_{mem} nearly linearly; model capacity shifts both clocks. Harder rules can therefore close the innovation window by pushing rule learning past memorization, while more training data can reopen it. We then examine what changes in diffusion models around these onsets (Sec. 4). At intermediate noise levels, denoising losses first distinguish rule-valid from rule-violating samples, and later distinguish training from held-out valid samples. Within this band, the learned landscape changes accordingly: attractor basins first form at Boolean vertices; rule-valid basins expand around τ_{rule}; and training-sample basins expand into neighboring valid regions around τ_{mem} (Fig. 3).

Finally, we test the scope of these findings (Sec. 5). Autoregressive models (e.g. GPT) show similar effects of G and N on the two clocks, although both τ_{rule} and τ_{mem} occur earlier than in the matched DiT experiments (Sec. 5.1). In rendered images, patch dimension adds a factor that can substantially delay rule learning; a fine-tuned Qwen3 model also exhibits the two clocks on a natural-language `Light-switch` task (Sec. 5.2). Similar dynamics appear for counting rules and categorical puzzles such as Sudoku (Sec. 5.3). Together, these results show how the timing of rule learning and memorization governs valid, novel generation, and how each transition is reflected in the evolving attractor landscape of diffusion models.

2 Generative modeling of parity: rule learning and memorization

We focus on group-level parity as a case study; extensions to other rules are discussed in Section 5.3.

Group-level parity Define $\mathcal{S}_d^+ = \{\mathbf{x} \mid \prod_{i=1}^d x_i = 1, x_i \in \{-1, 1\}\}$ to be the set of points with *even parity* on a d-dimensional boolean hypercube $\{-1, 1\}^d$ (Fig. 1**A**). For example, $\mathcal{S}_3^+ = \{(1, 1, 1), (-1, -1, 1), (-1, 1, -1), (1, -1, -1)\}$; note that $|\mathcal{S}_d^+| = 2^{d-1}$. Define $\mathcal{P}_d^+(\mathbf{x}) = |\mathcal{S}_d^+|^{-1} \sum_{\mathbf{y} \in \mathcal{S}_d^+} \delta(\mathbf{x} - \mathbf{y})$ to be the mixture of delta measures at all points of the set $\mathcal{S}_d^+$. The *group-level parity* is defined as $(\mathcal{S}_d^+)^m = \mathcal{S}_d^+ \times ... \times \mathcal{S}_d^+ \subset \{-1, 1\}^{md}$, where the d bits

in each of the m groups satisfy even parity, with $|(\mathcal{S}_d^+)^m| = (2^{d-1})^m$. Denote $(\mathcal{P}_d^+)^m$ the uniform measure over the rule-valid set $(\mathcal{S}_d^+)^m$, and define $\mathcal{U}_d$ as the uniform measure on d-dimensional Boolean hypercube.

Dataset design We construct samples $\mathbf{x} \in \mathbb{R}^D$ with $D = 36$. Each $\mathbf{x}$ is divided into D/G groups of size G that each satisfies even parity, i.e., $\mathbf{x} \in (\mathcal{S}_G^+)^{D/G}$. To generate $\mathbf{x}$, we first sample each group i.i.d. from $\sim \mathcal{P}_G^+$, and then concatenate the D/G groups to form $\mathbf{x}$ (Fig. 1**A**). The training set contains N unique samples obtained by rejection sampling, while individual groups may repeat across samples. The key design parameter for the dataset are hence D, G, N. For diffusion training, we reshape each sample to a 6×6 single channel image.

Model architecture We use Gaussian diffusion models and treat the dataset as lying in the continuous space $\mathbb{R}^D$. Specifically, we use the continuous-time EDM framework [Karras et al., 2022], x_0 prediction, and use diffusion transformer (DiT) [Peebles and Xie, 2023b] as the function approximator with EDM preconditioning. The baseline version is `DiT-mini` with 6 layers, 6 heads and hidden dimension 384, and we vary the model depth (in $\{3, 6, 12\}$) to examine the effect of model capacity. We use patch size 1 to maximize the capacity for modeling relation between bits. The DiTs are trained with Adam for 10^6 steps, without weight decay, with a constant learning rate 10^{-4} and batch size 256. Training details are deferred to App. B.1.

Evaluation Throughout training, we periodically generate samples using Heun's second-order deterministic sampler [Karras et al., 2022] and evaluate quantization, rule satisfaction, and memorization (details in App. B.3). We first measure proximity to the Boolean cube by $d_{\ell_\infty}(\mathbf{x}) = \max_i ||x_i| - 1|$, and call a sample invalid (quantization error) at threshold ϵ if $d_{\ell_\infty}(\mathbf{x}) > \epsilon$. After binarizing each element to $\{-1, 1\}$, we compute parity *accuracy* at both the group and sample levels (Fig. 1**A**). Group-level accuracy has chance level $1/2$, while sample-level accuracy requires all D/G groups to be correct and therefore has chance level $2^{-D/G}$ (Fig. 1**B**, black dashed).

We also report group- and sample-level *memorization ratios*, defined as the fractions of generated groups or full samples that exactly match those in the training set. If the model samples uniformly from the true parity-constrained distribution $(\mathcal{S}_G^+)^{D/G}$, the expected sample-level memorization ratio is $N/(2^{G-1})^{D/G} = N \cdot 2^{-\frac{G-1}{G}D}$. If it samples uniformly from the full Boolean cube, the corresponding baseline is $N \cdot 2^{-D}$.

General performance Using these metrics, we find that DiTs learn the parity-constrained distribution well only for small G. At the sample level, all models achieve near-perfect rule accuracy for easy rules $G \leq 4$, but the accuracy deteriorates as G increases; larger models extend this learnable regime but do not remove the degradation (Fig. 1**B**). Group-level evaluation shows the same trend more directly: parity accuracy remains near perfect for small G, but approaches the chance level 0.5 for larger G, especially in smaller models (Fig. 1**C**). Meanwhile, memorization takes up an increasing fraction of the rule-valid generations at larger G, indicating that apparent performance increasingly reflects exact retrieval of training examples rather than rule generalization.

Combinatorial creativity We find little evidence for novelty at the group level. For $G \leq 12$, most rule-valid generated groups are memorized training groups; for $G > 12$, the residual gap between group accuracy and group memorization is largely explained by chance parity satisfaction ($1/2$; Fig. 1**C**). Thus, novel valid samples arise primarily by recombining memorized groups into unseen full samples, reflecting a form of *combinatorial creativity*.

3 Two clocks of generative training

The above analysis, performed at the end of training, shows that increasing the rule complexity G shifts DiT generations from novel generalization toward memorization-dominated performance. But this static view does not explain *when* rule learning and memorization emerge during training, or *why* some settings produce novel valid samples while others do not. We therefore track rule accuracy and memorization over training and find two distinct timescales: τ_{rule}, when generated samples first become rule-valid, and τ_{mem}, when they begin to recall training samples. We call these the two *clocks* of generative training. Their gap defines an "innovation window", in which the model generates rule-valid but non-memorized samples.

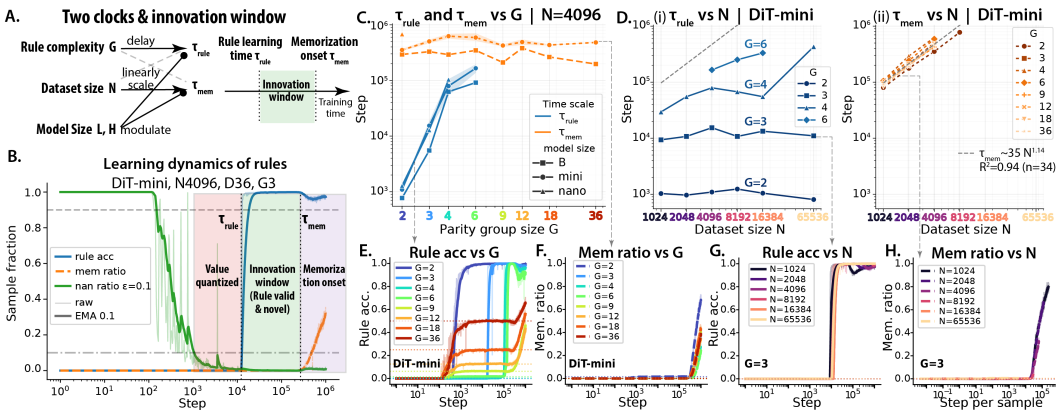


Figure 2: **Two clocks τ_{rule} and τ_{mem} scale differently with rule, data, and model. A.** Schematic: rule complexity G delays τ_{rule}, dataset size N scales τ_{mem} near-linearly, model size (L, H) modulates both. The interval $[\tau_{\text{rule}}, \tau_{\text{mem}}]$ is the *innovation window* of valid novel generation. **B.** Single-run learning dynamics (DiT-mini, N=4096, G=3): rule accuracy (blue), memorization ratio (orange dashed), and NaN/quantization error ratio with ϵ=0.1 (green); EMA over raw traces. Vertical dotted lines: τ_{rule} (first step rule-acc > 0.9) and τ_{mem} (first step mem-ratio > 0.1). Shaded phases: value quantization, innovation window, memorization onset. **C.** τ_{rule} (blue) and τ_{mem} (orange dashed) vs. G at N=4096 for three model sizes (B, mini, nano); τ_{rule} shown only for learnable G, i.e., $G \leq 6$. **D.** (i) τ_{rule} vs. N for $G \in \{2, 3, 4, 6\}$. (ii) τ_{mem} vs. N across all G, with power-law fit $\tau_{\text{mem}} \approx 35\, N^{1.14}$ (R^2=0.94, n=34). **E,F.** Rule accuracy and memorization ratio vs. step (DiT-mini, N=4096), one curve per G. **G,H.** Rule accuracy vs. step and memorization ratio vs. steps-per-sample (i.e. step $\times$ batch size / N), one curve per N (DiT-mini, G=3).

Two clocks in one run Let's follow a training run with DiT-mini, G=3, N=4096 (Fig. 2**B**). First, the invalid-value fraction rapidly decreases, indicating that the model has learned to sample from the Boolean cube. Next, the rule-learning transition τ_{rule} occurs, marked by a sharp rise in sample-level rule accuracy from chance to near-perfect performance. Memorization remains near baseline level at this point and starts climbing around the much later τ_{mem}, accompanied by a small uptick in the fraction of invalid samples and a small dip in rule accuracy. We operationally define τ_{rule} as the first step at which sample-level rule accuracy reliably exceeds 0.9, and τ_{mem} as the first step at which sample memorization fraction reliably exceeds 0.1. The interval $[\tau_{\text{rule}}, \tau_{\text{mem}}]$ is referred to as the *innovation window*, during which the model generations are rule-valid yet novel. This window of valid, novel generation is consistent with prior observations in diffusion models [Bonnaire et al., 2025], including random hierarchical grammar tasks [Favero et al., 2025], which focused on how the dataset size affects this window. Our setup differs from these prior work by providing more fine-grained control over the task complexity. Specifically, we next examine how rule complexity G, dataset size N, and model capacity differentially shift its two boundaries (Fig. 2**A**).

Determinants of rule-learning time τ_{rule} We find that *rule complexity monotonically delays τ_{rule}*. For small G, where the rule-learning transition occurs reliably, τ_{rule} increases from approximately 10^3 steps at $G = 2$ to 10^4 at $G = 3$ and 10^5 at $G = 4$; thus, each unit increase in G delays rule learning by roughly an order of magnitude (Fig. 2**C**, blue). At $G = 6$, the transition becomes unreliable, likely because τ_{rule} approaches τ_{mem} and the innovation window begins to close. For all model sizes, rules with $G \geq 9$ are not acquired within 10^6 steps, and their final accuracy can be fully attributed to the later memorization process, which matches the known difficulty of learning high-degree parity [Kearns, 1998, Barak et al., 2022, Abbe et al., 2024a]. In contrast, *τ_{rule} is only mildly modulated by dataset size N*: increasing N from 1024 to 65536 leads to indistinguishable rule learning curves at $G = 2, 3$ and mildly increases τ_{rule} for $G = 4, 6$ (Fig. 2**D**(i), **G**). Finally, *increasing model capacity accelerates rule learning τ_{rule} but does not remove the complexity barrier* [3]. Given a fixed G, larger DiTs have an earlier rule transition τ_{rule} (Fig. 2**C**, comparing DiT-B/mini/nano). However, even the largest 12-layer model we test fails at $G \geq 9$, showing the learnability frontier.

Determinants of memorization time τ_{mem} In contrast to τ_{rule}, *τ_{mem} is approximately invariant to the rule complexity G*: memorization occurs after a similar number of optimization steps regardless of the rule being acquired or not (Fig. 2**C**, orange; Fig. 2**F**). Rather, there is a *clear scaling of τ_{mem} in*

[3] See Appendix A for more discussions on related work.

the training set size N: increasing N shifts τ_{mem} nearly linearly, with empirical fit $\tau_{\text{mem}} \approx 35\, N^{1.14}$ ($R^2 = 0.94$, $n = 34$ across all G at DiT-mini scale; Fig. 2**D***(ii)*; see App. C.3 for scaling per G and model). This near-linear dependence agrees with prior results across image, language, and synthetic diffusion tasks [Bonnaire et al., 2025, Favero et al., 2025]. At our fixed batch size, memorization trajectories approximately collapse when training time is normalized by dataset size (step $\times$ batch / N), with onset around 10^4 in step-per-sample units (Fig. 2**H**). Finally, increasing model capacity also accelerates τ_{mem} (Fig. 2**C**, orange), with DiT-nano having nearly no memorization within 10^6 steps. This earlier onset also leads to higher memorization ratio, and higher memorization-driven accuracy at final evaluation (Fig. 1**B,C** across depths).

The innovation window These scaling relationships (Fig. 2**A**) together explain why models have a hard time learning higher G rules: at a fixed model size and N, higher G significantly scales up τ_{rule} while keeping τ_{mem} stable; thus, extrapolating this trend, the hypothetical τ_{rule} for higher G will hit the τ_{mem} ceiling, resulting in models only reaching the memorization solution and closing the innovation window (Fig. 2**C**). In contrast, increasing dataset size N delays τ_{mem} and opens up the innovation window for learning harder rules: for example, $G = 6$ is not learned at $N = 1024, 2048$ as memorization happens first, but it is learned for larger $N = 4k, 8k, 16k$ as τ_{mem} ceiling goes up (Fig. 2**D**(i)). In hierarchical grammar learning, increasing dataset size likewise delays memorization and enables valid, novel generation [Favero et al., 2025, Fig. 5]. Further, a higher model capacity shifts both τ_{rule} and τ_{mem} earlier, so it need not widen the innovation window (Fig. 2**C**).

4 Dissecting rule learning and memorization dynamics

Section 3 provides statistical observations for the rule-learning time τ_{rule} and memorization time τ_{mem}; but what underlying mechanisms govern them? This section provides an in-depth analysis of training dynamics from three complementary perspectives: (1) the samples per-seed (Fig. 3**A**), (2) the loss per noise scale, (Fig. 3**B,C**), and (3) the vector field and energy landscape (Fig. 3**D,E**).

These three views jointly present the following picture. The rule learning and memorization are associated with a critical noise scale of $\sigma \approx 1$ (Fig. 3**B**). First, the model learns to generate on the Boolean hypercube by turning each vertex into an attractor at these noise scales (Fig. 3**D**, step 14k). After rule learning transition, the rule-valid samples expand their attractor basins over the rule-violating ones (Fig. 3**D**, step 119k). Finally, after memorization onset, basins of the trained samples expand further and eventually dominate the landscape (Fig. 3**D,E**, step 492k). We next discuss each perspective in detail.

4.1 Per-sample training dynamics of rule learning and memorization

With a deterministic sampler, diffusion samples are endpoints of probability-flow ODE trajectories $d\mathbf{x}_\sigma/d\sigma = -(\mathbf{D}_\theta(\mathbf{x}_\sigma, \sigma) - \mathbf{x}_\sigma)/\sigma$ that depend continuously on the learned velocity field, and hence on the network parameters θ. Thus, the generated samples change continuously across training steps, and we can infer the properties of the underlying landscape from these dynamics. Starting from fixed initial noises, we track the corresponding generated samples across training steps. We classify the generations into four categories: (1) invalid values (quantization-ambiguous), (2) valid values but rule-violating, (3) rule-valid and novel, and (4) valid and memorized from training set.

We then visualize the state transition between these four categories as a function of training steps. As shown in the raster (Fig. 3**A**, bottom), the rule-violating to rule-valid transition is sharp and synced across all initial seeds, while the later memorization happens more gradually and stochastically. In the late phase, individual samples consist entirely of valid values, but can flicker between being rule-valid or rule-violating, or being valid-novel or valid-memorized. Modeling the Markov transition probability between the four categories in the late phase, we see that the rule-violating state is non-attractive where samples tend to flip back to valid quickly, while the valid-memorized state is sticky where samples have a higher probability to stay (Fig. 13).

During late training (10^5–10^6 steps), generated samples continue to move closer to the training set in nearest-neighbor Hamming distance, even before τ_{mem} and even when restricted to rule-valid, non-memorized generations (Fig. 14). This is consistent with the intuition that the training samples are increasingly "attractive" whose basins devour more samples originally converging elsewhere. A more detailed analysis of per-sample training dynamics process can be found in App. C.4.1.

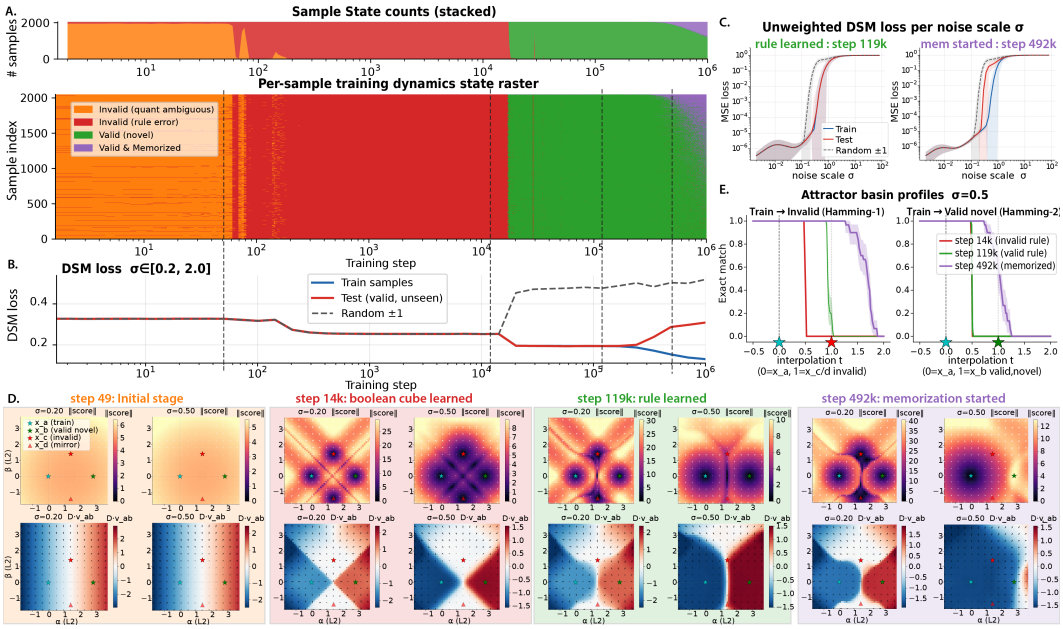


Figure 3: **Dissecting rule learning and memorization dynamics via multiple perspectives**. **A.** *Per-sample training-dynamics state raster* for 2000 fixed initial noises. Samples are classified into four states across training steps: invalid (quantization-ambiguous), invalid (rule-violating), valid & novel, and valid & memorized, shown as (*Top*) counts or (*Bottom*) per-seed evolution throughout training. The transition matrices are showed in Fig. 13. **B.** *DSM loss averaged over* $\sigma \in [0.2, 2.0]$, evaluated on training samples (blue), held-out rule-valid samples (red), and random Boolean-cube samples (dashed gray). **C.** *Unweighted DSM loss*, as a function of σ at (*left*) the rule-learning step (119k) and (*right*) the memorization-onset step (492k). All views show a sharp, synchronized rule-learning transition and a gradual, stochastic memorization onset. To understand why this happens, we provide the **D.** *vector-field and landscape view*, which shows (*top*) score magnitude $\|s_\theta\|$ and (*bottom*) projected denoising velocity $D - v_{ab}$ at $\sigma \in \{0.20, 0.50\}$. The visualization shows a projection on a 2D slice of the boolean hypercube containing four reference points (Section 4.3), where the basin of the training sample x_a keeps expanding during training. This is further supported by **E.** *Attractor basin profiles* at σ=0.5, which shows fractions of samples getting denoised to x_a when moving along a linear interpolations from x_a (α=0) to (*left*) invalid neighbors (i.e., x_c or x_d) or (*right*) valid neighbors (i.e., x_b): x_a and x_c, x_d are equally attractive basins in the landscape before rule learning (step 14k), and such symmetry is broken after rule learning (step 119k). x_a, x_b remain equally attractive at this time, but x_a becomes the dominating attractor after memorization onset (step 492k).

4.2 A loss perspective of rule learning and memorization

DiT training optimizes for the denoising score matching (DSM) objective $\mathcal{L}_\sigma = \mathbb{E}_{\mathbf{x}_0 \sim p_0} \|\mathbf{D}_\theta(\mathbf{x}_0 + \sigma\mathbf{z}, \sigma) - \mathbf{x}_0\|^2$ integrated over some noise scale $\int_\sigma \mathcal{L}_\sigma d\sigma$. Here, we break down the aggregation and track the DSM loss at individual noise scales across training steps (Fig. 3**B**,**C**, App. C.4.2).

We evaluate the DSM loss separately for I) training samples, II) held-out rule-valid samples, and III) random samples from the boolean hypercube, which allows to interpret what the learned score captures. Before the rule-learning transition, the DSM losses are indistinguishable between training samples and random Boolean samples (Fig. 3**B**, $\tau < 14$k), suggesting that the model is basically learning the score of the Boolean hypercube. After τ_{rule}, the DSM loss on training samples becomes lower than the hypercube (Fig. 3**B**, $\tau > 14$k). Further, around τ_{mem}, the DSM loss on training samples and held-out samples diverge from each other (Fig. 3**B**, $\tau > 492$k; Bonnaire et al. [2025]).

Notably, after τ_{mem}, the DSM loss separates between training and held-out samples only over an intermediate range of noise scales, $\sigma \in [0.2, 2.0]$ (Fig. 3**C**), recapitulate findings in Dodson et al. [2026]. This range matches the geometry of the Boolean hypercube: nearest Boolean vertices are separated by Euclidean distance 2, while nearest even-parity vertices are separated by $2\sqrt{2}$. Thus, memorization emerges when the denoising scale is comparable to the distance between neighboring data points, making it a property of the learned vector field at specific intermediate distances from the training samples. We next analyze this vector-field mechanism in detail in Section 4.3.

4.3 A vector field and landscape view of rule learning and memorization

Treating diffusion as a family of attractor dynamics [López Amado et al., 2026], we directly visualize the learned vector field at different training steps τ and noise scales σ (Fig. 3**D**), focusing on the intermediate noise band isolated by the loss analysis. We visualize a 2d slice of the learned high-dimensional vector field. The slice is chosen to pass through four Boolean vertices on a face of the Boolean hypercube: a training sample $\mathbf{x}_a$, a rule-valid held-out sample $\mathbf{x}_b$ at Hamming distance 2 from $\mathbf{x}_a$, and two rule-violating odd-parity samples $\mathbf{x}_c, \mathbf{x}_d$, each at Hamming distance 1 from both $\mathbf{x}_a$ and $\mathbf{x}_b$. The visualization (App. C.4.3) reveals the following two clear phenomena.

Rule learning transition Before rule learning, all four points are attractors with similar sized, symmetric basins (Fig. 3**D**, step 14k). After rule learning, suddenly, the attractor basins for rule-valid samples x_a, x_b expand sharply, shrinking the coverage of the basins of rule-violating samples x_c, x_d (Fig. 3**D**, step 119k). As a results, much fewer samples can fall on to the rule-violating corners of the cube. Note that at this moment, the basins of x_a, x_b are still roughly symmetric, consistent with the identical DSM losses on training and held-out samples (Fig. 3**C**, left). This configuration is stable for a long time, until the memorization onset.

Memorization onset The onset happens more gradually than the rule learning transition. At the onset, the basin of the training sample x_a continues to expand and invades the basin of x_b, as evidenced by the shift of the decision boundary towards x_b (Fig. 3**D**, step 492k). This is consistent with the post-τ_{mem} divergence of DSM loss between training and held-out samples (Fig. 3**B,C**): the learned vector field becomes a poorer denoiser for held-out samples, pointing toward the memorized training point $\mathbf{x}_a$ even when the noised samples are closer to $\mathbf{x}_b$. This attractor basin of x_a keeps growing and eventually devours the odd-parity samples x_c, x_d, as quantified by the basin-profile measured along 1d interpolation between $x_a \to x_c$ and $x_a \to x_b$ (Fig. 3**E**). Notably, this also explains the increasing invalid sample ratio in the late phase (Fig. 3**A**, Fig. 2**B**, late phase): while the trajectory points towards the memorized samples, a noised sample may accidentally pass by the rule-violating samples x_c, x_d along the trajectory, and get trapped by them as the noise scale lowers — at lower noise scales, the score is still very much that of the boolean hypercube, so every corner of the cube is still an attractor (Fig. 3**C**). This is also consistent with our previous observation that the late-phase invalid samples do not form a stable state, but keep flipping back to valid samples (Fig. 3**A**, bottom).

5 Generalization to broader settings

Our investigation has so far focused on diffusion model learning parity. This section shows that the observations therein hold more broadly across architectures and tasks.

5.1 Architecture generalization to autoregressive models

One may wonder whether observations in previous sections are specific to diffusion models or are a general property of generative models. We investigate this question by training GPT2-style transformers, whose architecture matches with DiT [4] but using autoregressive modeling objective on the same group-parity datasets. We provide hyperparameter ablation in Appendix C.8.

We find a similar two clocks structure for autoregressive models: for simpler rules $G \leq 6$, the GPT experiences a similar rule learning transition and a later memorization "transition" (Fig. 4**A**). Similar to diffusion (Fig. 2**A**), τ_{rule} for GPT is delayed by rule complexity G, while τ_{mem} is relatively consistent across G (Fig. 4**B**) and linearly modulated by the sample size N ($\tau_{\text{mem}} \approx 2.1N^{0.97}, R^2 = 0.94, n = 34$, GPT-mini, all G, Tab. 5, Fig. 12, 22). Unlike diffusion though, GPT learns and memorizes faster than DiT of a matching size, leading to a shorter "innovation window" in between (Fig. 4**A,B**). In addition, the transition to memorization is more sudden than DiTs (Fig. 4**A**): at $N = 4096$ GPT memorization rise to 1.0 in a few thousands of steps; in contrast, DiT memorization didn't saturate across 8×10^5 steps. We hypothesize that this is because autoregressive training repeatedly sees the same prefixes in training, whereas the noises in DiT training effectively serve as data augmentation that slows down both learning and memorization. This is also related to prior work's observation that diffusion can be more data-efficient [Prabhudesai et al., 2025, Ni et al., 2025].

[4] When having the same architecture, causal attention strictly weakens expressivity [Chen et al., 2025]. Our later discussion shows that GPT learns faster despite the capacity disadvantage.

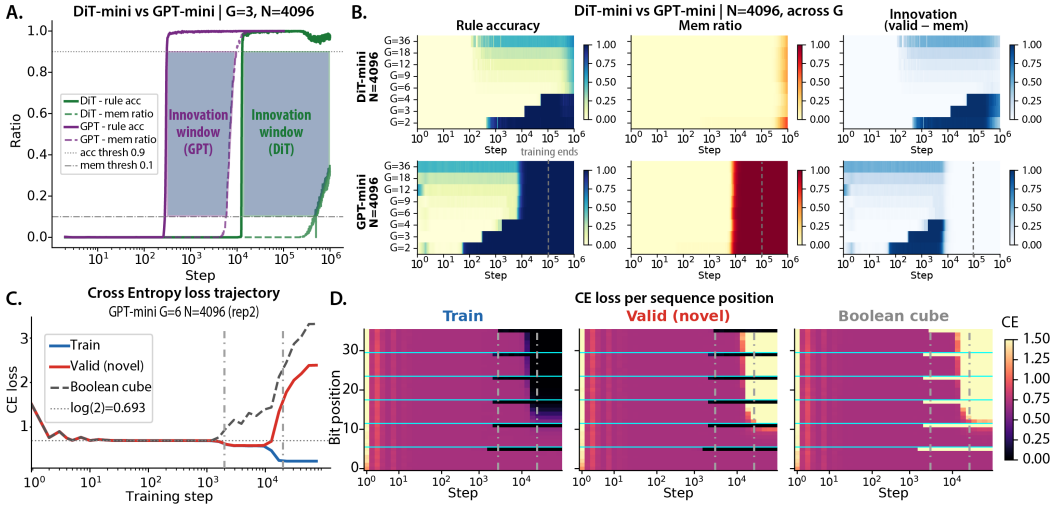


Figure 4: **Diffusion (DiT) and autoregressive (GPT) transformers share the two-clock structure with different timescales. A.** Sample-level rule accuracy (solid) and memorization ratio (dashed) vs. training step for DiT-mini (green) and GPT-mini (purple) at G=3, N=4096. Shaded: innovation window where rule accuracy > 0.9 and memorization < 0.1. **B.** Per-architecture heatmaps across $G \in \{2, 3, 4, 6, 9, 12, 18, 36\}$ at N=4096: rule accuracy (left), memorization ratio (middle), innovation rule_acc − mem_ratio (right). Top row: DiT-mini; bottom row: GPT-mini; vertical grey dashed line marks the GPT training cutoff. **C.** GPT-mini cross-entropy loss trajectory with G=6, N=4096 (run 2), on training samples (blue), held-out rule-valid samples (red), and random Boolean-cube samples (dashed grey). Reference: $\log(2) \approx 0.693$. **D.** GPT-mini per-bit-position cross-entropy heatmaps for the same three datasets (Train, Valid novel, Boolean cube). Cyan horizontal lines: G=6 group boundaries; vertical dashed lines: τ_{rule} and τ_{mem}.

Mechanistically, autoregressive models exhibit similar loss-split dynamics, as evidenced by the cross entropy loss for next-token prediction (NTP) against I) training samples, II) held-out rule-valid samples, and III) random boolean samples. The CE loss against random samples diverges from the other two at the rule learning transition; while the training and held-out samples losses diverge at the memorization onset time (Fig. 4**C**), mirroring the DSM-loss split observed in diffusion (cf. Fig. 3**B**). Interestingly, the two NTP loss divergences have different positional composition. For rule learning, the hypercube vs training losses diverge at the last bit in each group (Fig. 4**D**), i.e., $i \mod G = 0$, which are deterministic conditioned on the prefix. For memorization, the train-test divergence happens at all position past a certain position around 11 (Fig. 4**D**). This creates an intriguing parallel: learning and memorization manifest at specific noise scales for diffusion (Fig. 3**C**), and at specific token positions and prefix lengths in autoregressive models.

5.2 Toward more naturalistic image and language tasks

As a step toward more naturalistic settings, we express the same latent rules through rendered images and an everyday scenario described in natural language (Fig. 5**A**; App. B.4.3), relaxing the idealized bit representation. The image DiT models tokenize each RGB image into $P \times P$-pixel patches. In `Block`, D=36 bits form a 6×6 grid: each bit selects one of two fixed colors for a uniform square that fills one patch, giving one bit per token. `Orb` replaces each square with a shaded orb contained within the same patch, adding visual variation while preserving this correspondence. In `Flower`, D=24 bits determine the colors of two rings of twelve petals; each petal spans multiple patches, so recovering a bit requires combining information across tokens (Fig. 26). Representative images of all three tasks appear in Fig. 8. For language, the `Light-switch` task asks a fine-tuned Qwen3-1.7B to generate a 36-action T/F sequence satisfying group parity (prompt examples in Tab. 3). We decode generated images or parse language answers to test rule validity and training-set matches exactly.

Images. Similar to observations on binary images, higher-order parity remains harder and slower to learn: increasing G from 2 to 3 delays τ_{rule} from $2.2{\times}10^4$ to $4.3{\times}10^5$ steps on `Orb`, with similarly pronounced delays on `Flower` (Tab. 6), with higher G rule unattained in our training budget.

For images, an unexpected factor slowing down learning is the patch size, which corresponds to the model's input dimension per token d: an RGB patch of $P{\times}P$ pixels has $d = 3P^2$. At fixed token count, the `Block` sweep yields a clean scaling $\tau_{\text{rule}} \propto d^{0.66}$ for $d \in [1, 192]$, including the scalar

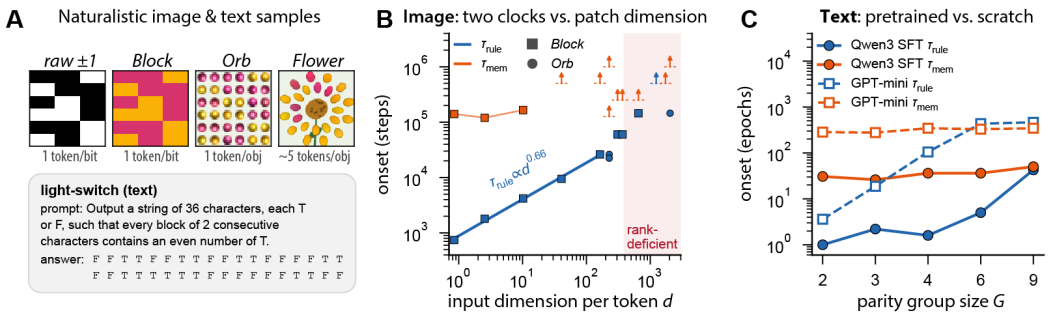


Figure 5: **Rule learning and memorization in more naturalistic image and language tasks. A.** Rendered latent bits and a `Light-switch` instruction. **B.** τ_{rule},τ_{mem} versus input dimension per token d, with D=T=36, N=2048, and parity G=2. Blue/orange denote rule/memorization; squares/circles distinguish `Block`/`Orb`. The fit only uses `Block` points $d \leq 384$ including the ± 1 anchor. Thin dashed boundaries with upward arrows indicate end of training and unobserved τ_{mem}. Shading marks $d > 384$, width of DiT. **C.** τ_{rule},τ_{mem} versus G, in epochs, for Qwen3-1.7B SFT and GPT-mini from scratch. Details see App. B.4.3 and Tab. 10.

d=1 reference (Fig. 5**B**; Tab. 7). At d=192, flat `Block` and shaded, nuisance-varied `Orb` have nearly identical onsets, suggesting a smaller effect for rendering details (Tab. 9). Thus, increasing image resolution through larger patches can slow rule learning even without adding tokens; the separate `Flower` sweeps show delays with both patch size and token count (Fig. 26; Tab. 8). These scaling suggests learning higher order rules underlying natural images can be even harder!

Memorization is also delayed: low-dimensional `Block` runs reach τ_{mem} at 1.2–1.7$\times 10^5$ steps, whereas no onset is observed for $d \geq 48$ within budget (Fig. 5**B**; Tab. 7). Several higher-dimensional runs therefore learn the rule while retaining an innovation window through the end of training.

Language. Fine-tuning Qwen3-1.7B reproduces the two-clock dynamics: rule learning precedes memorization, creating a window of valid, novel generation (Tab. 11). Increasing G primarily delays τ_{rule} while τ_{mem} varies less, narrowing the window to $\tau_{\text{mem}}/\tau_{\text{rule}} = 1.18$ at G=9 (Fig. 5**C**; Tab. 10). Qwen3 reaches the validity threshold in fewer epochs than GPT-mini trained from scratch, although model size and training setup also differ.

5.3 Rule generalization beyond parity

We start with a broader choice of binary rules which also require global coupling. Inspired by the counting problem in T2I diffusion [Mou et al., 2023], we consider the following rules: 1) *exact-K*, where each sample has exactly K bits positive; 2) *row-K*, where each row has exactly K bits positive; and more in App. C.6. Example training samples are shown in Fig. 6. Notably, exact-K and row-K rule has a rapid τ_{rule} comparable to parity with $G \in [2, 3]$; further their τ_{rule} is insensitive to the choice of K (Fig. 23**B**). Further, the memorization onset τ_{mem} is insensitive to the types of rules, showing robustness of our result.

Beyond learning binary rule, we experiment with Latin square and Sudoku (6x6), which are more natural and complex with multi-valued inputs (Fig. 7). We find DiTs can learn to generate substantial fraction of novel and valid samples (Fig. 24), but the max accuracy is farther from 1.0 than previous rules. We think the difficulty comes from the nestedness of constraints: even though each number uniqueness constraint is equivalent to row-$K = 1$, each entry is affected and constrained by multiple sets of rules (row/column/block). Consistently, lifting the column uniqueness constraint, the row-only Latin square can be learned rapidly, and accuracy rise to perfection within 1k steps (Fig. 25), thus the τ_{rule} is indeed comparable to row-K. A full investigation of these rules is deferred to future work.

An energy perspective for learning speed across tasks Above, we show that different rules exhibit markedly different learning times. A revealing contrast is parity versus *exact-K*: exact-K imposes a global constraint over all 36 bits, yet is learned as quickly as the easiest parity cases ($G = 2$), and its τ_{rule} is largely insensitive to K (Fig. 23**A,B**). Thus, globality alone does not determine rule-learning difficulty; the form of the constraint matters.

To see this, write a binary-rule energy as

$$E_d(\mathbf{x}) = \frac{1}{2}\sum_{i=1}^{d}(x_i^2-1)^2 + \lambda_p E_f(\mathbf{x}), \qquad \lambda_p > 0, \tag{1}$$

where the first term encourages $\mathbf{x}$ to lie on the Boolean cube and E_f encodes the rule. For the corresponding density $p_d(\mathbf{x};\beta) \propto \exp[-\beta E_d(\mathbf{x})]$, the low-temperature limit $\beta \to \infty$ concentrates on the rule-valid set $\mathcal{P}_d^+$, and the optimal score satisfies

$$\nabla \log p_d(\mathbf{x};\beta) = -\beta \nabla_{\mathbf{x}} E_d(\mathbf{x}), \qquad \nabla_{x_i} E_d(\mathbf{x}) = 2x_i(x_i^2-1) + \lambda_p \nabla_{x_i} E_f(\mathbf{x}). \tag{2}$$

The first, bit-local term explains the rapid learning of the Boolean hypercube (Fig. 2**B**); the second, rule-specific term accounts for the task-dependent learning difficulty.

For parity, one energy model reads $E_{G\text{-parity}}(\mathbf{x}) = \frac{1}{2}\sum_{i\in[d/G]}\left(\prod_{j\in[G]} x_{ij} - 1\right)^2$, whereas for exact-$K$, $E_{\text{exact-}K}(\mathbf{x}) = \frac{1}{2}\left(\sum_{i=1}^{d} x_i - K\right)^2$. Parity therefore requires degree-G multiplicative interactions in the rule-specific score, while exact-K is a low-degree count constraint. This is consistent with the known difficulty of learning high-degree parity-like functions [Abbe et al., 2023, 2024a].

6 Discussion

Rule learning and memorization follow distinct training clocks: rule complexity primarily delays τ_{rule}, dataset size delays τ_{mem}, and model capacity affects both. Across diffusion and autoregressive models, their relative timing determines whether an innovation window of valid, novel generation opens and whether a given rule can be learned. For diffusion models, both transitions leave distinct signatures at intermediate noise scales: around τ_{rule}, denoising losses separate rule-valid from random Boolean-cube samples as rule-valid attractor basins expand and invades those; around τ_{mem}, training and held-out losses diverge as training-sample basins encroach on neighboring valid regions.

Implications and scope. The difficulty of high-order parity is consistent with diffusion models' struggles on XOR-type relations in RAVEN [Wang et al., 2024]. Our rendered-image experiments further show that representation and tokenization can shift both clocks, while fine-tuned language models retain a transient innovation window. These extensions take a step toward more naturalistic settings, while generalization to unrestricted natural data remains open.

Directions for improved learning. To assist higher-order rule learning, candidate interventions include global communication through register tokens [Darcet et al., 2023], auxiliary representation objectives [Yu et al., 2024], and curricula that progressively increase rule complexity; their effects on both clocks remain to be tested. To alleviate memorization, larger datasets can delay τ_{mem} and early stopping within the innovation window is recommended. Our dissection also suggests a diagnostic: contrasting loss between valid and invalid samples with matched properties, and between valid training and held-out samples may help locate $\tau_{\text{rule}}, \tau_{\text{mem}}$ (Sec. 4.2).

Rule boundaries and negative feedback. Autoregressive and diffusion models learn from valid training examples without explicit negative examples identifying rule violations. Their objectives constrain the learned distribution but do not explicitly specify the boundary of the rule-valid support. The resulting generalization can be too broad, assigning probability to invalid configurations, or too narrow, concentrating on training examples while neglecting other valid configurations. These observations motivate testing explicit negative feedback through unlikelihood training [Welleck et al., 2020], preference-based fine-tuning [Wallace et al., 2024], or negative guidance during sampling [Koulischer et al., 2025] to better delineate rule-valid support.

Broader outlook. Our parity learning at intermediate-noise and boolean cube attractors at lower-noise suggest that, noise scales may expose different levels of rule structure. Whether the organization across noise levels tracks the hierarchy of rule complexity remains open. Our results on rule discovery and adherence in generative AI have potential implications for rule-faithful generation in structured and scientific domains.

Acknowledgments and Disclosure of Funding

B.W. and B.L. were supported by the Kempner Fellowship from the Kempner Institute at Harvard. This work was made possible in part by a gift from the Chan Zuckerberg Initiative Foundation to establish the Kempner Institute for the Study of Natural and Artificial Intelligence, and by the generous computing resources provided by the Kempner Institute and Harvard Research Computing. E.F. would also like to thank the CRISP Lab and the Calvin Coolidge Presidential Foundation for their support during her undergrad and A.M. as well as the Rhodes Trust for their support of her DPhil.

We thank the organizers of the NeurIPS 2025 Workshop on Structured Probabilistic Inference and Generative Modeling for accepting an earlier version of this work as an oral presentation, and for the helpful discussions it generated. We are grateful to T. Anderson Keller for discussions on the connections between various energy functions and their corresponding score architectures, which inspired our focus on the parity task. We thank Alex Damian for insightful discussions on the parametric energy model of parity and for pointing us to relevant literature on learning dynamics. Finally, we thank Shane Jiaqi Shang and Haim Sompolinsky for early discussions on rule learning in diffusion models and transformers — particularly regarding the RAVEN task — from which our work draws much of its inspiration.

The authors declare no competing interests.

References

E. Abbe, Enric Boix-Adserà, and Theodor Misiakiewicz. Sgd learning on neural networks: leap complexity and saddle-to-saddle dynamics. *Annual Conference Computational Learning Theory*, 2023. doi: 10.48550/arXiv.2302.11055.

Emmanuel Abbe, Samy Bengio, Aryo Lotfi, and Kevin Rizk. Generalization on the unseen, logic reasoning and degree curriculum. *Journal of Machine Learning Research*, 25(331):1–58, 2024a.

Emmanuel Abbe, Samy Bengio, Aryo Lotfi, Colin Sandon, and Omid Saremi. How far can transformers reason? the globality barrier and inductive scratchpad. *Neural Information Processing Systems*, 2024b.

Emmanuel Abbe, Elisabetta Cornacchia, Jan Hazla, and Donald Kougang-Yombi. Learning high-degree parities: The crucial role of the initialization. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=OuNIWgGGif.

B. Barak, Benjamin L. Edelman, Surbhi Goel, S. Kakade, Eran Malach, and Cyril Zhang. Hidden progress in deep learning: Sgd learns parities near the computational limit. *Neural Information Processing Systems*, 2022. doi: 10.48550/arXiv.2207.08799.

S. Bhattamishra, Arkil Patel, Varun Kanade, and Phil Blunsom. Simplicity bias in transformers and their ability to learn sparse boolean functions. *Annual Meeting of the Association for Computational Linguistics*, 2022. doi: 10.48550/arXiv.2211.12316.

Giulio Biroli, Tony Bonnaire, Valentin De Bortoli, and Marc Mézard. Dynamical regimes of diffusion models, 2024. URL https://arxiv.org/abs/2402.18491.

Tony Bonnaire, Raphaël Urfin, Giulio Biroli, and Marc Mézard. Why diffusion models don't memorize: The role of implicit dynamical regularization in training, 2025. URL https://arxiv.org/abs/2505.17638.

Sam Buchanan, Druv Pai, Yi Ma, and Valentin De Bortoli. On the edge of memorization in diffusion models. *Advances in Neural Information Processing Systems*, 38:96113–96157, 2025.

Abdulkadir Canatar, Blake Bordelon, and Cengiz Pehlevan. Spectral bias and task-model alignment explain generalization in kernel regression and infinitely wide neural networks. *Nature Communications*, 12(1):2914, May 2021. ISSN 2041-1723. doi: 10.1038/s41467-021-23103-1. URL https://doi.org/10.1038/s41467-021-23103-1.

Lijie Chen, Binghui Peng, and Hongxun Wu. Theoretical limitations of multi-layer transformer. *2025 IEEE 66th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 2631–2653, 2025. doi: 10.1109/FOCS63196.2025.00136.

Zhengdao Chen. On the interpolation effect of score smoothing in diffusion models. *arXiv preprint arXiv: 2502.19499*, 2025.

David Chiang and Peter Cholak. Overcoming a theoretical limitation of self-attention. *arXiv preprint arXiv: 2202.12172*, 2022.

Alex Damian, Jason D. Lee, and Joan Bruna. The generative leap: Sharp sample complexity for efficiently learning gaussian multi-index models. *arXiv preprint arXiv: 2506.05500*, 2025.

Timothée Darcet, Maxime Oquab, Julien Mairal, and Piotr Bojanowski. Vision transformers need registers. *arXiv preprint arXiv:2309.16588*, 2023.

Nick Dodson, Xinyu Gao, Qingsong Wang, Yusu Wang, and Zhengchao Wan. Two calm ends and the wild middle: A geometric picture of memorization in diffusion models. *arXiv preprint arXiv:2602.17846*, 2026.

Benjamin L. Edelman, Surbhi Goel, Sham Kakade, Eran Malach, and Cyril Zhang. Pareto frontiers in neural feature learning: Data, compute, width, and luck. *arXiv preprint arXiv: 2309.03800*, 2023.

Alessandro Favero, Antonio Sclocchi, and Matthieu Wyart. Bigger isn't always memorizing: Early stopping overparameterized diffusion models. *arXiv preprint arXiv:2505.16959*, 2025.

Emma Finn, T. Anderson Keller, Manos Theodosis, and Demba E. Ba. Origins of creativity in attention-based diffusion models, 2025. URL https://arxiv.org/abs/2506.17324.

Margalit Glasgow. SGD finds then tunes features in two-layer neural networks with near-optimal sample complexity: A case study in the XOR problem. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=HgOJlxzB16.

Michael Hahn. Theoretical limitations of self-attention in neural sequence models. *Transactions of the Association for Computational Linguistics*, 8:156–171, 2020.

Michael Hahn and Mark Rofin. Why are sensitive functions hard for transformers? *arXiv preprint arXiv: 2402.09963*, 2024.

Yujin Han, Andi Han, Wei Huang, Chaochao Lu, and Difan Zou. Can Diffusion Models Learn Hidden Inter-Feature Rules Behind Images?, February 2025.

John J. Vastola. Generalization through variance: How noise shapes inductive biases in diffusion models, April 2025.

Zahra Kadkhodaie, Florentin Guth, Eero P Simoncelli, and Stéphane Mallat. Generalization in diffusion models arises from geometry-adaptive harmonic representation. In *The twelfth international conference on learning representations*, 2024. URL https://openreview.net/forum?id=ANvmVS2Yr0.

Mason Kamb and Surya Ganguli. An analytic theory of creativity in convolutional diffusion models, 2024. URL https://arxiv.org/abs/2412.20292.

Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. *arXiv preprint arXiv:2206.00364*, 2022.

Michael Kearns. Efficient noise-tolerant learning from statistical queries. *J. ACM*, 45(6):983–1006, November 1998. ISSN 0004-5411. doi: 10.1145/293347.293351. URL https://doi.org/10.1145/293347.293351.

Juno Kim and Taiji Suzuki. Transformers provably solve parity efficiently with chain of thought. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=n2NidsYDop.

Felix Koulischer, Johannes Deleu, Gabriel Raya, Thomas Demeester, and Luca Ambrogioni. Dynamic negative guidance of diffusion models. In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2410.14398.

Sixu Li, Shi Chen, and Qin Li. A good score does not lead to a good generative model, 2024a. URL https://arxiv.org/abs/2401.04856.

Xiang Li, Yixiang Dai, and Qing Qu. Understanding generalizability of diffusion models requires rethinking the hidden gaussian structure. *Advances in neural information processing systems*, 37: 57499–57538, 2024b.

Bingbin Liu, Jordan T Ash, Surbhi Goel, Akshay Krishnamurthy, and Cyril Zhang. Transformers learn shortcuts to automata. *arXiv preprint arXiv:2210.10749*, 2022.

Cristina López Amado, Marco Fumero, and Francesco Locatello. From modes to memories: Characterizing the scale-space dynamics of diffusion models, 2026. URL https://arxiv.org/abs/2609.39648.

Artem Lukoianov, Chenyang Yuan, Justin Solomon, and Vincent Sitzmann. Locality in image diffusion models emerges from data statistics. *Advances in Neural Information Processing Systems*, 38:95121–95157, 2026.

Andrea Montanari and Pierfrancesco Urbani. Dynamical decoupling of generalization and overfitting in large two-layer networks. *arXiv preprint arXiv: 2502.21269*, 2025.

Chong Mou, Xintao Wang, Liangbin Xie, Jing Zhang, Zhongang Qi, Ying Shan, and Xiaohu Qie. T2i-adapter: Learning adapters to dig out more controllable ability for text-to-image diffusion models. *AAAI Conference on Artificial Intelligence*, 2023. doi: 10.48550/arXiv.2302.08453.

Jinjie Ni, Qian Liu, Longxu Dou, Chao Du, Zili Wang, Hang Yan, Tianyu Pang, and Michael Qizhe Shieh. Diffusion language models are super data learners. *arXiv preprint arXiv: 2511.03276*, 2025.

Matthew Niedoba, Berend Zwartsenberg, Kevin Murphy, and Frank Wood. Towards a mechanistic explanation of diffusion model generalization. *arXiv preprint arXiv:2411.19339*, 2024.

William Peebles and Saining Xie. Scalable Diffusion Models with Transformers, March 2023a.

William Peebles and Saining Xie. Scalable Diffusion Models with Transformers, March 2023b. URL http://arxiv.org/abs/2212.09748. arXiv:2212.09748 [cs].

Alethea Power, Yuri Burda, Harri Edwards, Igor Babuschkin, and Vedant Misra. Grokking: Generalization beyond overfitting on small algorithmic datasets. *arXiv preprint arXiv:2201.02177*, 2022.

Mihir Prabhudesai, Mengning Wu, Amir Zadeh, Katerina Fragkiadaki, and Deepak Pathak. Diffusion beats autoregressive in data-constrained settings. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=W5Ht05jF4c.

Qwen Team. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Itamar Shoshani and Ohad Shamir. Hardness of learning fixed parities with neural networks. *arXiv preprint arXiv: 2501.00817*, 2025.

Bhavya Vasudeva, Deqing Fu, Tianyi Zhou, Elliott Kau, Youqi Huang, and Vatsal Sharan. Simplicity bias of transformers to learn low sensitivity functions. *arXiv preprint arXiv: 2403.06925*, 2024.

Bram Wallace, Meihua Dang, Rafael Rafailov, Linqi Zhou, Aaron Lou, Senthil Purushwalkam, Stefano Ermon, Caiming Xiong, Shafiq Joty, and Nikhil Naik. Diffusion model alignment using direct preference optimization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024. URL https://arxiv.org/abs/2311.12908.

Binxu Wang and Cengiz Pehlevan. An Analytical Theory of Spectral Bias in the Learning Dynamics of Diffusion Models, March 2025.

Binxu Wang and John Vastola. The unreasonable effectiveness of gaussian score approximation for diffusion models and its applications. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=I0uknSHM2j.

Binxu Wang, Jiaqi Shang, and Haim Sompolinsky. Diverse capability and scaling of diffusion and auto-regressive models when learning abstract rules, November 2024. URL http://arxiv.org/abs/2411.07873. arXiv:2411.07873.

Binxu Wang, Jacob Zavatone-Veth, and Cengiz Pehlevan. A random matrix theory perspective on the consistency of diffusion models. *arXiv preprint arXiv:2602.02908*, 2026.

Sean Welleck, Ilia Kulikov, Stephen Roller, Emily Dinan, Kyunghyun Cho, and Jason Weston. Neural text generation with unlikelihood training. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=SJeYe0NtvH.

Kaiyue Wen, Huaqing Zhang, Hongzhou Lin, and Jingzhao Zhang. From sparse dependence to sparse attention: Unveiling how chain-of-thought enhances transformer sample efficiency. *International Conference on Learning Representations*, 2024. doi: 10.48550/arXiv.2410.05459.

Shunyu Yao, Binghui Peng, Christos Papadimitriou, and Karthik Narasimhan. Self-attention networks can process bounded hierarchical languages. *arXiv preprint arXiv:2105.11115*, 2021.

Sihyun Yu, Sangkyung Kwak, Huiwon Jang, Jongheon Jeong, Jonathan Huang, Jinwoo Shin, and Saining Xie. Representation alignment for generation: Training diffusion transformers is easier than you think. *arXiv preprint arXiv:2410.06940*, 2024.

Huijie Zhang, Jinfan Zhou, Yifu Lu, Minzhe Guo, Liyue Shen, and Qing Qu. The emergence of reproducibility and consistency in diffusion models. *arXiv preprint arXiv:2310.05264*, 2023.

Contents

1 Introduction **1**

2 Generative modeling of parity: rule learning and memorization **2**

3 Two clocks of generative training **3**

4 Dissecting rule learning and memorization dynamics **5**

4.1 Per-sample training dynamics of rule learning and memorization 5

4.2 A loss perspective of rule learning and memorization 6

4.3 A vector field and landscape view of rule learning and memorization 7

5 Generalization to broader settings **7**

5.1 Architecture generalization to autoregressive models 7

5.2 Toward more naturalistic image and language tasks 8

5.3 Rule generalization beyond parity 9

6 Discussion **10**

A Related work **16**

B Experimental Details **18**

B.1 DiT architecture and training 18

B.2 Compute resources 18

B.3 Sampling and evaluation metrics 18

B.4 Task definitions and dataset generation 19

B.4.1 Exact-K and row-K 19

B.4.2 Latin square and Sudoku 21

B.4.3 More naturalistic image and language tasks 23

B.5 GPT architecture and training 26

C Extended Results and Additional Analyses **28**

C.1 Extended evaluation results for DiT-Parity at training endpoint 28

C.2 Extended DiT learning curves 29

C.3 Extended memorization-time scaling for DiT and GPT 31

C.4 Dissecting rule learning and memorization dynamics 33

C.4.1 Per-sample training dynamics 33

C.4.2 DSM loss by noise scale 36

C.4.3 Vector-field and attractor-basin analyses 39

C.5 Autoregressive vs diffusion supplementary results 44

C.6 Beyond Parity Rules 45

C.7 Extended results for more naturalistic images and language 48

C.7.1 Image results 48
C.7.2 Language results 51
C.8 Optimization ablations 52

D Broader Impacts **53**

E LLM Usage **54**

A Related work

Rule learning in Diffusion models Prior work has demonstrated that diffusion models learn diverse rules with mixed success. Wang et al. [2024] showed that unconditional diffusion models can learn to generate according to only some of the rules in RAVEN's progression matrices encoded as integer arrays. In particular, rules corresponding to logical operations (AND, OR, XOR) over sets of attributes were difficult to learn. Similarly, Han et al. [2025] examined rule learning in the pixel space, showing that diffusion models can learn the coarse proportional relationship between bars and shadows length but not the precise rule specified in the training set. These prior works prompt this study, which examines what kind of binary rules can be learned.

Memorization and Creativity in Diffusion models The question of when diffusion models are able to generate genuinely novel samples is relevant both for scientific understanding and for preventing these models from reproducing potentially sensitive training data. From the rule learning perspective, the model that truly learns the rule should not simply regurgitate the training set, but learn the data manifold underlying it. From a score-matching perspective, however, exactly matching the score of the empirical data distribution implies that the reverse process will simply reproduce training samples [Kamb and Ganguli, 2024, Li et al., 2024a, Wang and Vastola, 2024]. Yet high-quality diffusion models routinely generate images that are not identical copies of any training example, raising the question of what mechanisms enable this generalization.

A striking empirical demonstration of generalization comes from Kadkhodaie et al. [2024] and Zhang et al. [2023]: when two diffusion models are trained on disjoint splits of the same dataset, both models—given sufficient data—produce highly consistent samples from matched noise seeds, and these samples differ from the nearest training images in either split. This cross-split consistency indicates that the models converge onto a shared generative structure (i.e. data manifold) rather than memorizing their respective training sets.

To understand why diffusion models generalize despite having sufficient memorization capacity, prior work has studied inductive biases coming from the model architectures. One line of work attributes generalization to the fact that well-trained diffusion models behave surprisingly similarly to a *linear-score diffusion model*, which captures only the first two moments of the data distribution [Wang and Vastola, 2024, Li et al., 2024b]. Under this view, cross-split consistency follows naturally from the largely shared Gaussian summary statistics of the two splits, which induce a shared linear score structure [Wang et al., 2026]. Beyond linear structures, Kamb and Ganguli [2024] show that when the score network is a simple convolutional neural network, its locality and translation equivariance favor patch-wise composition, producing globally novel samples that remain locally consistent "mosaics." Lukoianov et al. [2026] further demonstrate that the effective receptive field governing this locality is itself determined by image statistics and the linear (Wiener-filter) score. Wang and Pehlevan [2025] generalize this picture, showing that different architectural constraints induce different distributional approximations: linear networks learn the Gaussian approximation, while circular convolutional networks learn the stationary Gaussian process approximation. Finn et al. [2025] extend the analysis to attention-based architectures, providing evidence that a final self-attention layer promotes global consistency across distant regions, organizing locally plausible features into coherent global layouts.

Our work follows from the recent line of work viewing generalization and memorization as different phases of diffusion model training, where the phases are mediated by both architectural mentioned above and training-dynamical inductive biases [Biroli et al., 2024, Bonnaire et al., 2025, Buchanan et al., 2025, Chen, 2025, Favero et al., 2025, J. Vastola, 2025]. Most related to us are Bonnaire et al. [2025] and Favero et al. [2025], which also report two separated but related time scales for

generalization and memorization as well as a linear relation between dataset size and the memorization onset time. In comparison, our work focuses on rule learning with a tractable underlying distribution, which enables both a clear delineation between generalization and memorization and mechanistic understanding of the training process, while accounting for multiple factors including task complexity, dataset and model sizes. It is worth-noting that the timescale separation between generalizable learning and overfitting exists more generally in machine learning. In particular, Montanari and Urbani [2025] has identified the same timescale separation in feature learning of a 2-layer neural network, where they characterize the time as a function of the network width under a fixed overparameterization ratio.

Learning parity Many results in this paper focus on parity learning, a versatile testbed that has been widely adopted for understanding both the representational and learning aspects of neural networks [Hahn, 2020, Bhattamishra et al., 2022, Glasgow, 2024, Abbe et al., 2024a, 2025, Shoshani and Shamir, 2025]. The difficulty depends on the number of bits that the parity is defined over, where more bits require a higher boolean sensitivity or larger weight norms in the case of neural networks. For Transformers specifically, learning parity requires growing the MLP norms [Liu et al., 2022, Hahn and Rofin, 2024] and the use of normalization layers [Hahn, 2020, Yao et al., 2021, Chiang and Cholak, 2022]. Even when a network is sufficiently expressive, parity is computationally challenging to learn [Kearns, 1998, Barak et al., 2022, Edelman et al., 2023, Wen et al., 2024, Kim and Suzuki, 2025]. In this work, we explore parity learning from a generative modeling perspective, leveraging this well studied problem to characterize how much modern generative modeling framework, in particular diffusion, can learn these underlying structures.

Score complexity and spectral bias. As discussed in Section 5.3, the parity score naturally decomposes into a local term and a global multiplicative term $\prod_{j=1}^{G} x_j$, whose polynomial degree grows with G. In the Fourier/Walsh-Hadamard basis, higher-degree interactions correspond to higher-frequency components, and it is well established that neural networks exhibit a *spectral bias*, fitting low-frequency components before high-frequency ones [Canatar et al., 2021, Wang and Pehlevan, 2025]. This framework offers a natural explanation for our learning dynamics: small-G components emerge early in training, while large-G components appear only much later—if they appear at all. When the latter are not learned from data, accuracy improvements in late training tend to come from memorization rather than genuine rule acquisition. A more formal score-complexity analysis could help predict the point at which models shift from generalizing to overfitting, and explain how architectural constraints shape this transition.

Larger group size complicates learning It is well known that higher-degree polynomials (i.e., larger G) is more computationally challenging to learn, where the required number of steps grows exponentially in the polynomial degree [Barak et al., 2022, Abbe et al., 2023, Damian et al., 2025]. We observe similar phenomenon in our experiments (Fig. 2**C,E**), where rules with higher G tend to require more gradient steps to learn. For transformer specifically, it has also been observed that higher-degree or more global functions are harder to learn [Bhattamishra et al., 2022, Abbe et al., 2024b, Hahn and Rofin, 2024, Vasudeva et al., 2024].

More depth eases learning Although one layer suffices to express the parity function for a fixed G [Hahn, 2020, Liu et al., 2022], we observe that a greater depth leads to better learning empirically. One intuition is that representing the optimal score function of the ground-truth parity distribution requires the network to reach certain Lipschitzness [Hahn, 2020], and using more layers means that the required per-layer Lipschitzness [5] is smaller, which can be easier to reach in learning. This intuition is also consistent with our finding that rule learning often precedes memorization, since the Lipschitzness of the memorizing score function is usually higher than that of the score function of the ground truth rule distribution; the former requires a larger weight norm, as also noted in Montanari and Urbani [2025]. Further, it is consistent with our observation that deeper DiTs tend to memorize more (Fig. 1**B,C**), as depth facilitates the growth of Lipschitzness.

[5] The Lipschitzness comes from both the weight norms and the scaling introduced by normalization layers [Yao et al., 2021, Hahn and Rofin, 2024].

B Experimental Details

B.1 DiT architecture and training

Architecture. We use a 2d Diffusion Transformer (DiT) [Peebles and Xie, 2023a] for images. The input $\mathbf{x} \in \{-1, +1\}^D$ (D=36) is reshaped to a 6×6 single-channel image and tokenized with patch size 1, yielding 36 tokens each of dimension 1. Tokens are linearly embedded to the hidden dimension and combined with 2D sinusoidal positional embeddings. The noise level σ is embedded via a Fourier feature projection followed by a two-layer MLP; these timestep embeddings modulate each transformer block via adaptive layer norm (adaLN). There is no class conditioning (`num_classes=0`) and the model does not predict the noise variance (`learn_sigma=False`).

Model scales used in this work are summarized in Table 1. The primary model for all analyses is DiT-mini; DiT-nano, DiT-S, and DiT-B are used for model-scale ablations.

Model	d_{model}	Depth	Heads	#Params
DiT-nano	192	3	3	∼2.2M
DiT-mini	384	6	6	∼16.5M
DiT-S	384	12	6	∼32.5M
DiT-B	768	12	12	∼129.5M

Table 1: DiT model scales. All use `mlp_ratio=4`, `patch_size=1`.

EDM preconditioning. We adopt the EDM preconditioning of Karras et al. [2022]. Given a noisy input $\tilde{\mathbf{x}} = \mathbf{x} + \mathbf{n}$, $\mathbf{n} \sim \mathcal{N}(0, \sigma^2 I)$, the network F_θ produces a denoised estimate via:

$$D_\theta(\tilde{\mathbf{x}}, \sigma) = c_{\text{skip}}(\sigma)\, \tilde{\mathbf{x}} + c_{\text{out}}(\sigma)\, F_\theta(c_{\text{in}}(\sigma)\, \tilde{\mathbf{x}},\ c_{\text{noise}}(\sigma))\,, \tag{3}$$

where $c_{\text{skip}} = \sigma_{\text{data}}^2/(\sigma^2 + \sigma_{\text{data}}^2)$, $c_{\text{out}} = \sigma\sigma_{\text{data}}/\sqrt{\sigma^2 + \sigma_{\text{data}}^2}$, $c_{\text{in}} = 1/\sqrt{\sigma_{\text{data}}^2 + \sigma^2}$, and $c_{\text{noise}} = \ln(\sigma)/4$. We set σ_{data}=1.0 (consistent with data in $\{-1, +1\}^D$).

Loss and noise schedule. We use the EDM denoising score matching (DSM) objective:

$$\mathcal{L} = \mathbb{E}_{\mathbf{x},\sigma,\mathbf{n}}\left[\lambda(\sigma)\, \|D_\theta(\mathbf{x} + \mathbf{n}, \sigma) - \mathbf{x}\|^2\right], \tag{4}$$

with loss weight $\lambda(\sigma) = (\sigma^2 + \sigma_{\text{data}}^2)/(\sigma\, \sigma_{\text{data}})^2$. Noise levels are drawn from a log-normal distribution: $\ln \sigma \sim \mathcal{N}(P_{\text{mean}},\, P_{\text{std}}^2)$ with P_{mean}=−1.2, P_{std}=1.2 (same as the EDM paper defaults for continuous data). This distribution concentrates training at intermediate noise levels ($\sigma \approx e^{-1.2}$≈0.3) while providing coverage from σ_{min}=0.002 to σ_{max}=80.

Optimizer and training schedule. All DiT experiments use Adam (β_1=0.9, β_2=0.999) with learning rate η=10^{-4}, batch size 256, and 10^6 gradient steps. Weight decay is 0 for baseline runs; AdamW with decoupled weight decay is used only in the optimization ablation (§C.8). No learning rate schedule or warmup is applied. Training data is sampled once before training and held fixed throughout; the model sees each sample approximately $\lfloor 10^6 \times 256/N \rfloor$ times.

B.2 Compute resources

Experiments used NVIDIA H100 and A100 GPUs. The rendered-image and Qwen3 fine-tuning runs each used one H100 with 80 GB memory. Table 2 separates the previously reported synthetic-run estimates from the additional rendered-image and language experiments. For the latter, we sum per-run Weights & Biases runtimes, falling back to the span of TensorBoard training timestamps for five interrupted image runs. The rendered-image accounting includes all 36 recorded runs, including five with partial training logs; the language accounting includes the five corrected-prompt SFT runs used for the reported learning curves.

B.3 Sampling and evaluation metrics

Throughout training, at increasingly larger spacing of steps, we generate samples from a fixed set of 2048 initial noises, and evaluate them according to the following criterion. We used Heun's 2nd order deterministic sampler with 35 steps from $\sigma_{max} = 80.0$ to $\sigma_{min} = 0.002$ [Karras et al., 2022].

Model / data	Steps per run	Hours per run	Runs	GPU-hours
DiT-nano / synthetic	10^6	$\sim$9	8	$\sim$72
DiT-mini / synthetic	10^6	$\sim$8	102	$\sim$816
DiT-B / synthetic	10^6	$\sim$48	12	$\sim$576
GPT-nano	10^5	$\sim$1	8	$\sim$8
GPT-mini	10^5	$\sim$0.8	45	$\sim$37
GPT-B	10^5	$\sim$5	16	$\sim$80
DiT-mini / rendered	4×10^4–1.5×10^6	0.7–23.3	36	227.3
Qwen3-1.7B / SFT	16,000	3.0–3.1	5	15.2
Total			**232**	**$\sim$1832**

Table 2: Compute accounting, approximately 76 GPU-days in total. Synthetic entries retain the original estimates for 191 runs (approximately 1590 GPU-hours), based on representative TensorBoard or checkpoint timestamps; the added image and SFT entries sum measured per-run runtimes. Rendered step counts give the range of configured budgets, including partial runs. The total is a lower-bound accounting of the research effort: it excludes additional exploratory synthetic runs, earlier SFT attempts, standalone zero-shot evaluations and offline analyses, and the original pretraining of Qwen3.

First, we evaluate how far the samples are from the boolean cube $\{-1,1\}^D$, as measured by the ℓ_∞ distance $d_{\ell_\infty}(\mathbf{x}) = \max_i ||x_i| - 1|$. We call a sample *invalid* when $d_{\ell_\infty}(\mathbf{x}) > \epsilon$, and calculated the invalid fraction for various thresholds $\epsilon \in \{0.1, 0.01\}$.

Next, we binarize each element of the sample to $\{-1,1\}$ and evaluate the parity of the binarized sample $\bar{\mathbf{x}}$, at both group and sample level (Fig. 1**A**). The group parity accuracy is defined over a group of G elements, with chance level 2^{-1}. For the sample parity accuracy, a prediction is correct only if the parities of all D/G groups are correct, for which the chance level is $2^{-D/G}$ (Fig. 1B, dashed black).

Further, we examine the *memorization ratios* of groups and samples, which are the fractions of generated groups or samples that coincide with those in the training dataset. If the model learns the true distribution (i.e., the uniform measure) on $(\mathcal{S}_G^+)^{D/G}$, then the sample memorization ratio will be $N/(2^{G-1})^{D/G} = N \cdot 2^{-\frac{G-1}{G}D}$ (Fig. 1B, dashed red); and group memorization ratio will be roughly $\min(2^{G-1}, N \cdot D/G)/2^{G-1}$, while the specific value depends on the uniqueness of training groups, so it's computed empirically (Fig. 1C, dashed red). If the model learns the uniform measure on the entire boolean cube, then the sample memorization will be $N \cdot 2^{-D}$ (Fig. 1B, dashed blue); and group memorization ratio will be roughly $\min(2^{G-1}, N \cdot D/G)/2^G$, with empirical values showed (Fig. 1C, dashed blue).

B.4 Task definitions and dataset generation

B.4.1 Exact-K and row-K

The binary rule datasets all use the same 6×6 image format, but differ in which subsets of bits are coupled by the constraint. This shared visual format lets us compare rule-learning and memorization dynamics without changing the model architecture or sampler.

Exact-K rule. A binary vector $\mathbf{x} \in \{-1,+1\}^D$ satisfies the *exact-K* rule if exactly K entries equal $+1$:

$$\sum_{i=1}^{D} \mathbf{1}[x_i = +1] = K. \tag{5}$$

The valid set has size $\binom{D}{K}$. For D=36 this ranges from $\binom{36}{3}$=7,140 (K=3) to $\binom{36}{18}\approx9.1\times10^9$ (K=18). Crucially, the constraint can be written as a single quadratic penalty $(\sum_i x_i - (2K-D))^2$, so the score (gradient) couples all bits through a *degree-1* global sum. This contrasts with group parity, whose score involves a degree-$(G-1)$ product, making exact-K far easier to learn.

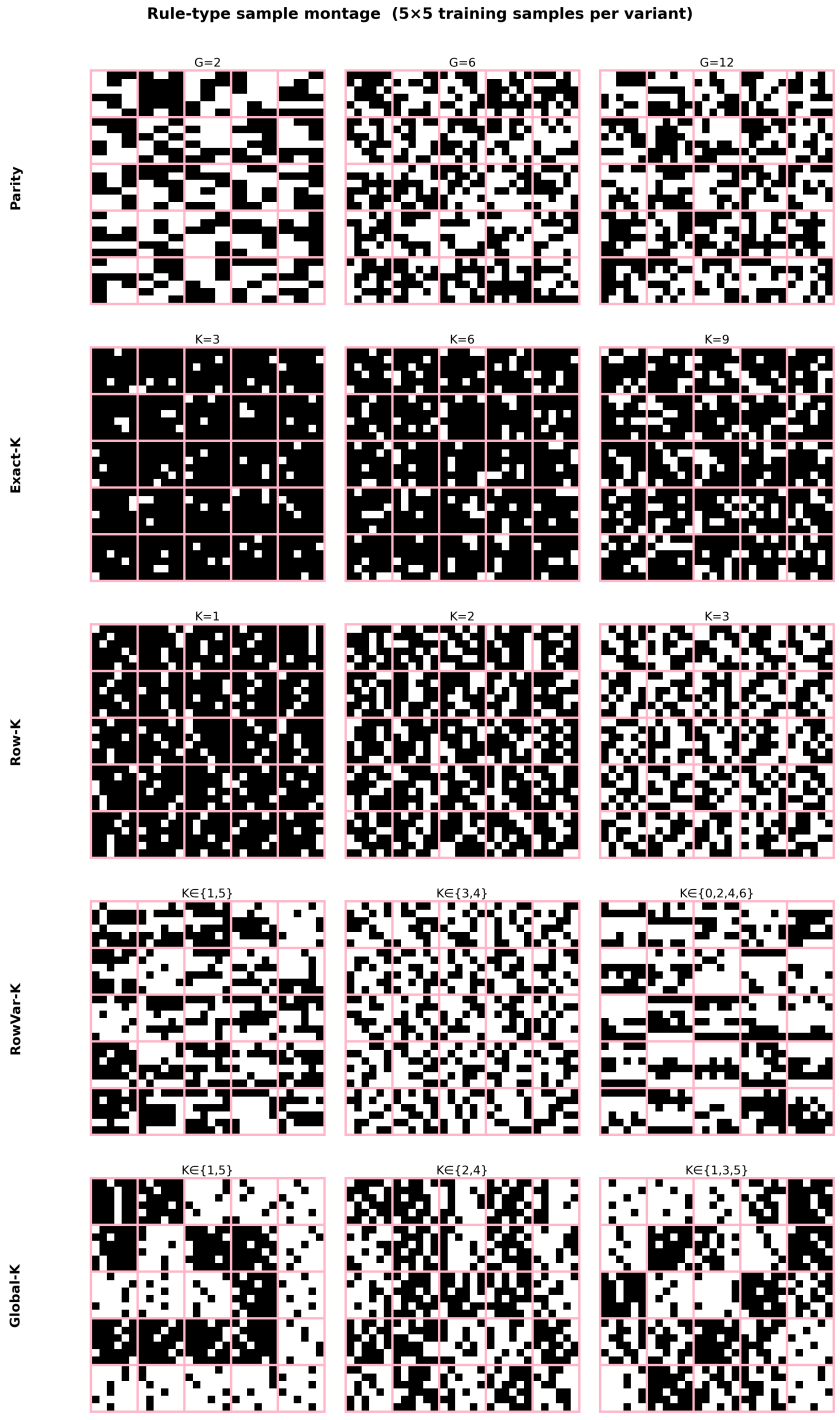


Figure 6: **Training-sample montage for binary rule families.** Each panel shows a 5×5 grid of representative 6×6 training samples for one rule variant. Rows correspond to group parity, exact-K, row-K, row-variable-K, and global-K datasets. Columns vary the group size G, exact count K, or allowed count set $\mathcal{K}$. The montage illustrates how the same binary image space supports qualitatively different constraints: parity imposes high-order group interactions, exact-K imposes a global count, row-K imposes local row counts, row-variable-K allows each row to choose from a set, and global-K shares a single count choice across all rows in a sample.

Exact-K evaluation. After binarizing generated samples to $\{-1, +1\}$ (threshold ϵ=0.1), a sample is *rule-valid* if its count of $+1$ entries equals exactly K. Sample accuracy and memorization ratio are computed identically to the parity case (§B.3), with statistical memorization baseline $N/\binom{D}{K}$.

Exact-K experiments. We train DiT-mini (D=36, N=4096) for $K \in \{3, 4, 6, 8, 9, 12, 18\}$. All use the same protocol as the parity experiments (lr= 10^{-4}, batch= 256, 10^6 steps, EDM sampler with 35 steps). Across all K values, a clear two-phase rule-then-memorize transition is observed, with rule learning occurring orders of magnitude faster than for comparable parity tasks, consistent with the lower-degree score argument.

Row-K rule. We additionally study constraints on an $n\times n$ grid (n=6, D=36). Three constraint variants are considered:

- **Row-K (fixed):** every row has exactly K entries equal to $+1$. Valid set: $\binom{n}{K}^n$.
- **Row-variable-K:** each row independently draws its count from a fixed set $\mathcal{K}$. Valid set: $\left(\sum_{k\in\mathcal{K}}\binom{n}{k}\right)^n$.
- **Global-K:** a single $K \in \mathcal{K}$ is drawn per sample, shared across all rows. Valid set: $\sum_{k\in\mathcal{K}}\binom{n}{k}^n$.

These variants isolate whether the model learns per-row structure, or must additionally infer which K applies globally.

Row-K evaluation. A generated grid is rule-valid if every row has the correct count of $+1$ entries ($= K$ for fixed, $\in \mathcal{K}$ for variable/global). Sample accuracy is the fraction of grids where *all* rows satisfy the constraint simultaneously.

Row-K experiments. We train DiT-mini (n=6, N=4096) on: fixed $K \in \{2, 3\}$; variable-K with $\mathcal{K} \in \big\{\{1, 5\}, \{3, 4\}, \{0, 2, 4, 6\}, \{3, 4, 5, 6\}\big\}$. Scalar-encoded ($[-1, +1]$) and one-hot (with zero-mean normalization) representations are both evaluated. As with exact-K, all row-K variants learn the rule in far fewer steps than parity, confirming that degree-1 per-row sums are easily captured by the DiT score.

B.4.2 Latin square and Sudoku

Latin square rule. An $n\times n$ *Latin square* is a grid filled with n distinct symbols $\{0, \ldots, n-1\}$ such that each symbol appears exactly once in every row and exactly once in every column. The number of valid $n\times n$ Latin squares is exactly $12, 576, 161{,}280, 812{,}851{,}200$ for $n = 3, 4, 5, 6$. The row and column uniqueness constraints require each symbol to appear exactly once in each row and column, placing Latin squares structurally between row-K (row-only, single-symbol) and more complex combinatorial rules.

Latin square row only rule We also considered a easier rule than Latin square, i.e. only considering the row uniqueness rule, namely, each row is a permutation of n, thus the number of valid $n\times n$ row only Latin squares samples is $(n!)^n$, i.e. $216,\ 331{,}776,\ 24{,}883{,}200{,}000,\ 139{,}314{,}069{,}504{,}000{,}000$ for $n = 3, 4, 5, 6$.

Sudoku rule. A 6×6 *Sudoku* (blocks 2×3) adds a third uniqueness constraint: each of the six 2×3 sub-blocks must also contain all six symbols exactly once. This reduces the valid set from $812{,}851{,}200$ (Latin square) to $28{,}200{,}960$ ($\approx 3.5\%$ of Latin squares satisfy the block constraint).

Encoding. Two encodings are evaluated: (i) **scalar** — integer symbols linearly mapped to $[-1, +1]$ (for n=6: $\{0, \ldots, 5\} \mapsto \{-1.0, -0.6, \ldots, +1.0\}$, input tensor shape (n, n)); (ii) **one-hot** — each cell encoded as a length-n binary vector (input tensor shape (n, n, n)), optionally zero-mean normalized.

Evaluation. For scalar encoding, generated samples are snapped to the nearest valid symbol (ϵ=0.15) and verified for row, column, and (for Sudoku) block uniqueness. A sample is rule-valid only if all constraints are simultaneously satisfied. For one-hot encoding, samples are decoded to integer grids before the same checks are applied. Memorization ratio is computed as for parity: the fraction of generated grids that exactly match a training-set grid.

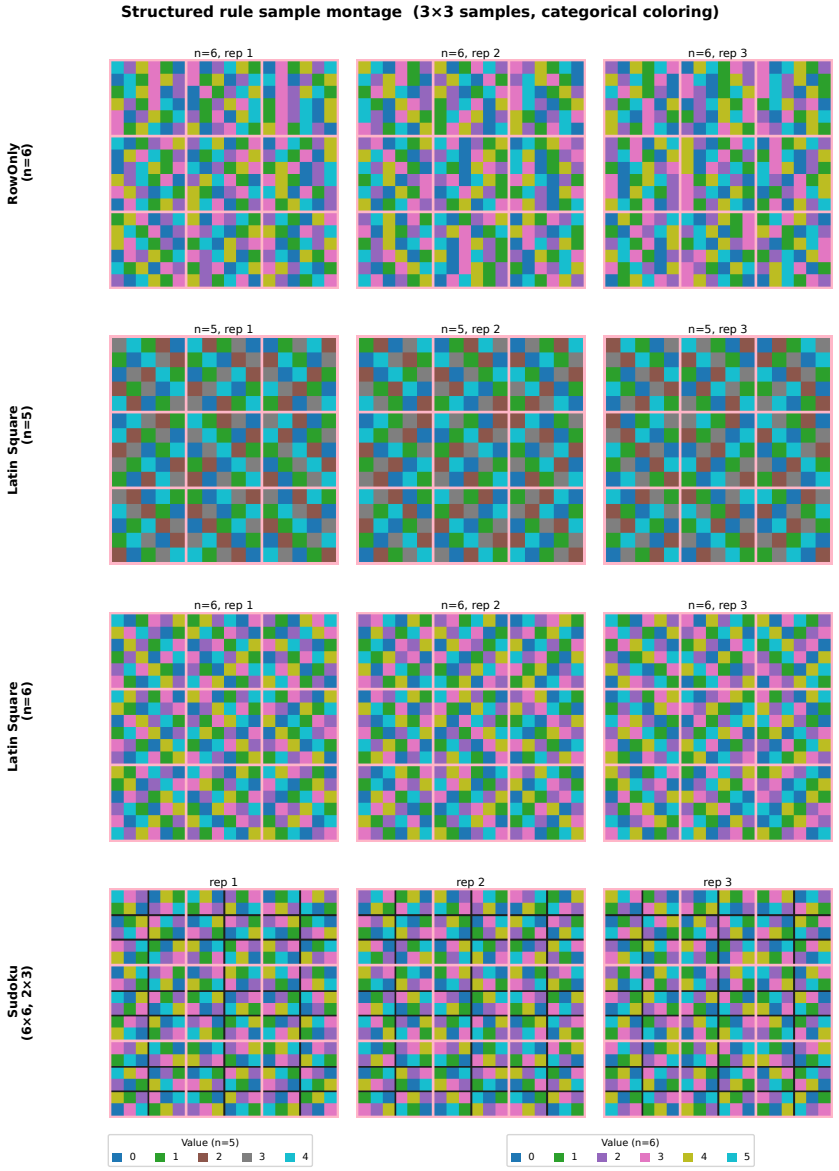


Figure 7: **Training-sample montage for structured categorical rule families.** Each row shows representative training samples from a different structured rule family, with three independently drawn samples shown per family. Rows correspond to RowOnly Latin Square ($n = 6$), Latin Square ($n = 5$), Latin Square ($n = 6$), and Sudoku (6×6 grid with 2×3 blocks). Colors denote categorical values: five colors for $n = 5$ and six colors for $n = 6$. The montage illustrates how the same grid-based categorical image space supports increasingly structured constraints, from row-wise value constraints to Latin-square row/column constraints and Sudoku-style row, column, and block constraints.

Experiments. We train DiT-mini and DiT-B on $n{=}5$ and $n{=}6$ Latin squares ($N{=}4096$, both encodings), and DiT-mini on $6{\times}6$ Sudoku ($N{=}4096$, scalar and zero-mean one-hot). Latin squares are sampled via cyclic-base permutation (rows, columns, symbols), covering approximately 46% of the $n{=}6$ isotopy classes; Sudoku grids are obtained by rejection-sampling Latin squares (approximately $29\times$ oversampling required). Both tasks exhibit a clear rule-learning onset followed by a later memorization onset, demonstrating that the two-clock phenomenology extends to multi-valued, combinatorially structured rules.

B.4.3 More naturalistic image and language tasks

The synthetic benchmarks assign each bit to a scalar token, leaving open how the two-clock dynamics change when a rule must be recovered from visual objects or natural-language instructions. In the following, we vary these representations while retaining the latent rules and exact validity tests. We describe the setup below, and present corresponding results in App. C.7.

Underlying rules. We use group parity as defined in Sec. 2 and exact-K as defined in App. B.4.1, retaining their notation and valid-support sizes. Each dataset contains N distinct valid latent sequences of D bits; rendering changes their visual appearance, while the same latent constraints determine correctness.

Image encoding and dataset generation. For RGB images of side length S, patch dimension is $d{=}3P^2$ and token count is $T{=}(S/P)^2$. The three datasets progressively relax the scalar-bit representation and the alignment between objects and tokens.

`Block`**.** We arrange $D{=}36$ bits on a $6{\times}6$ grid and map the two bit values to two fixed RGB colors. Each cell is a uniform $P{\times}P$ square that fills exactly one patch token for transformer, with no background, shading, or nuisance variation. Varying $P \in \{1, 2, 4, 8, 11, 12, 16, 24\}$ changes the number of pixel values per token while preserving token number $T{=}36$ and one bit per token. A separate single-channel $P{=}1$ anchor uses the raw ±1 values to check agreement with the original synthetic pipeline. This dataset isolates the effect of *patch dimensionality* using varying image resolution at fixed latent structure and object–token alignment.

`Orb`**.** The same $6{\times}6$ latent grid is rendered as colored, shaded ellipsoids on a background, with diffuse shading and specular highlights. Photometric nuisance varies lighting direction, object color, background tint, and pixel noise; geometric nuisance, where enabled, varies size, shape, orientation, and position. Each object is checked on the rendered pixels to remain within its own patch, preserving one object per token. The main rule-comparison runs use 48-pixel images, $P{=}8$ ($d{=}192$, $T{=}36$), and fixed 5.8-pixel geometry with photometric nuisance. A separate 4.8-pixel variant allows geometric jitter, while the rendering ablations also compare flat and shaded objects with and without nuisance. These comparisons introduce within-token visual structure while retaining the `Block` dataset's object–token correspondence.

`Flower`**.** We map a $2{\times}12$ latent grid to two rings of twelve petals, with petal color carrying the bit and ring and angular position specifying its index. The renderer varies global rotation, scale, and position, as well as individual petal geometry, color, shading, background tint, and sensor noise, using supersampled anti-aliasing. Unlike the `Block` and `Orb` datasets, petal boundaries are not aligned to transformer patches, so recovering a bit requires combining evidence across tokens. At 96 pixels and $P{=}8$, an image has $T{=}144$ tokens of dimension $d{=}192$, and a petal overlaps 4.8 tokens on average, measured from decoder blob labels. The scaling sweeps change image resolution and patch size together, or hold patch size fixed, to vary patch dimension and token count separately.

For all three datasets, latent sequences and their rendered nuisance realizations are sampled once and held fixed throughout training; nuisance is not resampled on each presentation. Representative training images are shown in Fig. 8.

Image evaluation. We decode generated images back to latent sequences and apply the rule checks and sample/group accuracy definitions of Apps. B.3 and B.4.1. A sample is valid only when the image is structurally readable and the recovered sequence satisfies the rule; unreadable images remain in the denominator and count as failures. Group accuracy follows the latent group checks, with incomplete latent vectors counted as failures; the additional image-structure check also applies to sample accuracy. We validated the decoder on held-out rendered images by comparing the recovered

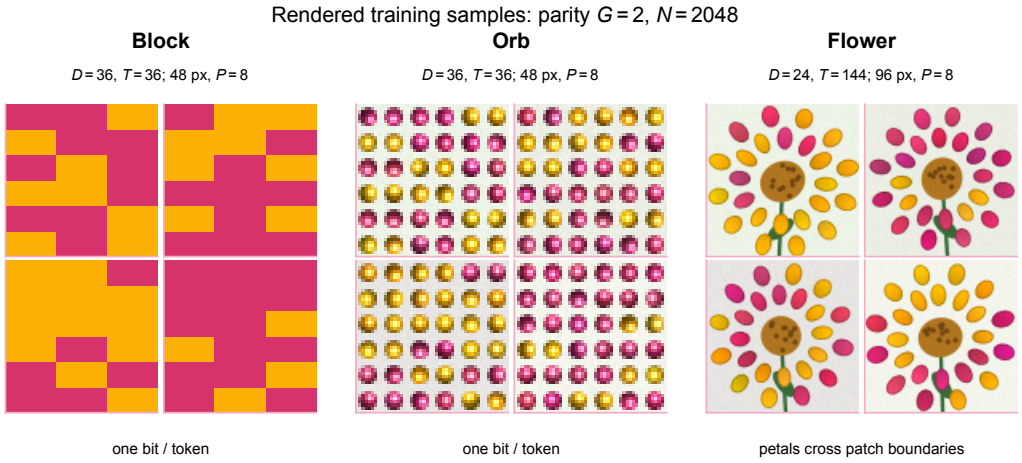


Figure 8: **Training-sample montage for rendered datasets.** Each dataset shows four saved training images in a 2×2 arrangement, all with parity $G{=}2$ and $N{=}2048$. `Block` and `Orb` share the same $D{=}36$ latent sequences in corresponding positions, with one bit per token ($T{=}36$, 48-pixel images, $P{=}8$). `Flower` encodes $D{=}24$ bits in petal colors ($T{=}144$, 96-pixel images, $P{=}8$); petals cross patch boundaries. Images are embedded at their original training resolution. Patch-grid examples for `Flower` appear beneath the scaling plots in Fig. 26.

bit at every object position with its known latent value. For the `Flower` dataset, we additionally changed the color of one petal at a time, creating a single-bit violation of the original rule. In all tested cases, the decoder recovered the modified latent sequence correctly, and the subsequent rule check rejected the corrupted sample.

Image experiments. All rendered runs use DiT-mini (6 layers, 6 heads, width $h{=}384$, MLP ratio 4), EDM, Adam, learning rate 10^{-4}, batch size 256, and $N{=}2048$. The training noise distribution and 35-step Heun sampler follow Apps. B.1 and B.3; σ_{data} is the training-pixel standard deviation (1.0 for the raw anchor). Budgets vary by run, reaching $1.5{\times}10^6$ steps. We normally evaluate 1024 samples at log-spaced checkpoints supplemented by linear spacing late in training. The `Block` ladder fixes $D{=}T{=}36$ and varies P; the `Flower` sweeps vary d at fixed $T{=}144$, or T at fixed $d{=}192$. The synthetic exact-K references use $D{=}36$, $N{=}4096$, whereas `Flower` uses $D{=}24$, $N{=}2048$; these cross-domain ratios are descriptive comparisons. There is one training run per configuration.

Language `light-switch` task and encoding. We also tested rule learning in an everyday scenario described in natural language. In the `Light-switch` task, $D{=}36$ people pass through a room in sequence, each choosing whether to flip the switch once (`T`) or leave it unchanged (`F`). The light starts off and must be off again after each consecutive, non-overlapping group of G people. Equivalently, every group contains an even number of `T`s; identifying `T` with -1 and `F` with $+1$ recovers the parity convention of Sec. 2. We evaluate zero-shot performance at $G \in \{2, 3, 4, 6, 9, 12, 18, 36\}$ using a bare instruction (d0, 35 words) and three narrative formulations (d1–d3, 52, 99, and 177 words). For the fine-tuning experiments, we focus on $G \in \{2, 3, 4, 6, 9\}$, training on d0 and evaluating on all four prompt formulations. The four prompt texts are shown in Tab. 3. Answers use spaces between letters so each bit occupies one `Qwen3` token. The checker parses a complete 36-letter answer and tests parity; strict format compliance is recorded separately. Validity is the fraction of all completions with a recoverable rule-valid answer.

Language experiments and evaluation. We fully fine-tune `Qwen3-1.7B` [Qwen Team, 2025] on 4096 distinct valid answers to the fixed d0 prompt, with loss on the 36 answer tokens and EOS only. Training uses learning rate 10^{-5}, effective batch 128, bf16, and 500 epochs (16,000 steps; approximately three H100 hours). Evaluation uses the token-checked raw training prefix, temperature 1.0, top-p 0.95, and a 108-token cap. At 54 roughly log-spaced checkpoints (plus start and the end), we draw 1024 d0 completions and 256 for each unseen narrative prompt. Zero-shot evaluation uses the `intruct` model's chat template, with 128 draws per rule–prompt cell. For comparison with GPT-mini, we report epochs sB/N at optimizer step s and batch size B (128 for Qwen3, 256 for GPT-mini).

Dressing	Exact prompt text
`d0` Bare instruction 35 words	*Output a string of 36 characters, each `T` or `F`, such that every block of 2 consecutive characters contains an even number of `T`. Write the 36 letters separated by single spaces, and nothing else.*
`d1` Short narrative 52 words	*A light starts off and 36 people each may flip its switch once, in order; write `T` for flip and `F` for no flip so that the light is off again after every group of 2 consecutive people has passed through. Write the 36 letters separated by single spaces, and nothing else.*
`d2` Full narrative 99 words	*A room contains a single light controlled by one switch. Flipping the switch toggles the light: on becomes off, off becomes on. Thirty-six people enter the room one at a time, and each person may flip the switch at most once before leaving. The light is off before the first person enters, and the light is off again after every group of 2 consecutive people has passed through. Write the thirty-six actions in order, using `T` if that person flips the switch and `F` if they do not. Write the 36 letters separated by single spaces, and nothing else.*
`d3` Narrative with distractors 177 words	*A room contains a single light controlled by one switch. Flipping the switch toggles the light: on becomes off, off becomes on. Thirty-six people enter the room one at a time, and each person may flip the switch at most once before leaving. The light is off before the first person enters. The people, in order, are Ana, Ben, Cleo, Dev, Eve, Femi, Gus, Hana, Ivan, Jo, Kit, Lena, Milo, Nia, Omar, Pia, Quinn, Rosa, Sam, Tara, Uma, Vik, Wren, Xu, Yuki, Zane, Ada, Bo, Cass, Dara, Eli, Fay, Gil, Hugo, Ines, Jae. The room also contains a bookshelf, a wooden chair, and a window that stays shut throughout. Nobody speaks, and the door closes on its own after each visit. Ignore everything except the switch: what matters is only that the light is off again after every group of 2 consecutive people has passed through. Write the thirty-six actions in order, using `T` if that person flips the switch and `F` if they do not. Write the 36 letters separated by single spaces, and nothing else.*

Table 3: **Four text dressings of the `Light-switch` task.** Verbatim prompts for $D{=}36$, parity $G{=}2$, and space-separated answers, generated by the experimental prompt renderer. SFT uses d0; d1–d3 are unseen narrative formulations used to evaluate transfer of the same rule. Word counts include the output-format instruction. SFT and transfer use the raw text shown here without a chat template.

Memorization and novel-valid generation. We measure memorization by exact training-set matches of decoded latent bits from images or text (instead of pixel). Naturalistic-image and language onsets use the fraction of valid outputs that match training examples, whereas the synthetic parity and GPT analyses use the fraction of all outputs. The chance baseline b is the training-set size divided by the number of rule valid sequences, corresponding to uniform sampling from the rule valid support (App. B.3). Novel-valid yield is validity multiplied by one minus the conditional training-match fraction.

Onset times and threshold setting. We select thresholds for rule learning and memorization separately, then apply the same onset-extraction procedure to both metrics.

Rule-learning threshold. We use sample validity $a_{\text{rule}} = 0.5$ across domains. This threshold crossing, denoted τ_{rule}, establishes rule generalization only when accompanied by substantial novel-valid generation: reproducing training examples also raises validity. For exact-K, this exceeds the independent-bit baseline with positive-bit probability K/D (approximately 0.14–0.21 for the synthetic references). For language, it captures the main validity transition without requiring subsequent fluctuations to remain above 0.9. `Flower` parity $G{=}3$ does not reach sample validity 0.5 within budget; for $G{=}3, 4$, we therefore report group-validity onset at 0.75, with matching synthetic reference thresholds. These group-level onsets measure partial rule acquisition rather than whole-sample validity.

Memorization threshold. We detect over-concentration on training examples relative to the uniform-valid-support baseline b, using

$$a_{\text{mem}} = \min\{0.9, \max(2b, 3/1024)\}.$$

The twice-baseline criterion requires an excess over chance training matches. The fixed rate floor $3/1024$ prevents isolated rare matches from triggering onset when b is very small; it is retained for GPT's 2048-draw evaluations and does not correspond to a literal three-match requirement for conditional ratios. The cap accommodates tasks with large training-set coverage: for `Flower` parity

G=2, $b = 0.5$, and the threshold becomes 0.9. Because the two onsets use different metrics and thresholds, $\tau_{\text{mem}} < \tau_{\text{rule}}$ does not imply memorization followed by generalization; the later validity crossing may be driven by training-set reproduction.

Robust onset extraction. To reduce sensitivity to isolated checkpoint fluctuations, we first apply a centered width-3 running median to each metric trajectory, padding endpoints by repetition. Writing the smoothed observations as $\widetilde{z}_i$ at checkpoints s_i, we define onset as the first checkpoint from which the metric remains at or above its threshold through the final observation:

$$\tau(z; a) = \min\{s_i : \widetilde{z}_j \geq a \text{ for all } j \geq i\}. \tag{6}$$

Median smoothing suppresses isolated excursions, while the persistence requirement excludes temporary threshold crossings. If no such checkpoint exists, onset is not observed within budget. This retrospective criterion differs from the main analyses' consecutive-crossing definition and depends on the observed training horizon and checkpoint spacing. All reference onsets in App. C.7 are recomputed using this procedure.

Third-party assets. The Qwen3 model and tokenizer [Qwen Team, 2025] are obtained from the official model repository under Apache-2.0; the checkpoint identifier is `Qwen/Qwen3-1.7B`. Our implementations adapt the original DiT [Peebles and Xie, 2023b] and EDM [Karras et al., 2022] code, whose source repositories use CC BY-NC 4.0 and CC BY-NC-SA 4.0, respectively. These assets are used for noncommercial academic research; all task data are procedurally generated rather than collected from external datasets.

B.5 GPT architecture and training

Architecture. We train three GPT model variants at different scales to assess model-size dependence, summarized in Table 4. The default model used for all detailed analyses in this paper is GPT-mini.

Model	d_{model}	Layers	Heads	#Params (approx.)
GPT-nano	384	3	6	∼11M
GPT-mini	384	6	6	∼22M
GPT-B	768	12	12	∼86M

Table 4: GPT model variants used in parity learning experiments.

GPT-mini matches the scale of the DiT-mini used for diffusion experiments. We additionally sweep dataset sizes $N \in \{4096, 8192, 16384, 32768\}$ for GPT-mini, and train GPT-nano and GPT-B at N=4096 and N=32768 to verify that the two-phase rule-then-memorize dynamics are robust across model capacity and data scale.

Tokenization. The binary parity sequences $\mathbf{x} \in \{\pm 1\}^D$ are first mapped to $\{0, 1\}^D$, then a start-of-sequence (SOS) token with index 2 is prepended, yielding integer sequences of length D+1=37 from a vocabulary of size 3 (0, 1, SOS). The model is trained with next-token prediction cross-entropy loss, so at each position $k \in \{1, \ldots, D\}$ it predicts the k-th bit given the SOS token and bits 1 through $k-1$.

Training setup. All GPT-mini baseline experiments use AdamW with learning rate η=10^{-4}, weight decay λ=0.01, batch size 256, and 10^5 gradient steps. We save 40 checkpoints spaced uniformly in log-step (from step ≈500 to 10^5). At each checkpoint, N=2048 sequences are sampled autoregressively at temperature 1.0 and decoded back to $\{\pm 1\}^D$; rule accuracy and memorization ratio are computed identically to the DiT evaluation (§B.3).

Cross-entropy analysis across dataset splits. To dissect the rule-learning and memorization phases in the GPT, we evaluate the next-token prediction cross-entropy (CE) at each checkpoint on three held-out splits of N=4096 sequences each:

1. **Training set**: the N=4096 samples seen during training.
2. **Valid-novel test set**: rule-valid samples drawn from the same group-parity distribution, verified to not appear in the training set (Hamming distance ≥ 1 to every training sample).

3. **Boolean-cube samples**: uniformly random $\{0,1\}^D$ binary sequences with no parity structure, representing an unstructured null distribution.

The valid-novel CE first *decreases* when the model internalizes the parity rule, then *increases* again as memorization progresses and the model overfits the training distribution—a signature of grokking-then-forgetting dynamics. The boolean-cube CE serves as a ceiling reference; its divergence from the valid-novel curve marks the onset of rule learning.

Per-position cross-entropy. We additionally track the per-position CE: $\ell_k = -\mathbb{E}[\log p_\theta(x_k \mid x_1, \ldots, x_{k-1})]$ for $k \in \{1, \ldots, 36\}$ on the valid-novel split. Positions whose index is a multiple of the group size G (i.e., $k \equiv 0 \pmod{G}$) are the *last position in each group*; given the preceding $G-1$ bits of the group, this position is fully determined by the parity constraint. Consequently, its CE is expected to collapse to 0 first as the model learns the rule, providing a high-resolution probe of which structural positions are acquired earliest.

C Extended Results and Additional Analyses

C.1 Extended evaluation results for DiT-Parity at training endpoint

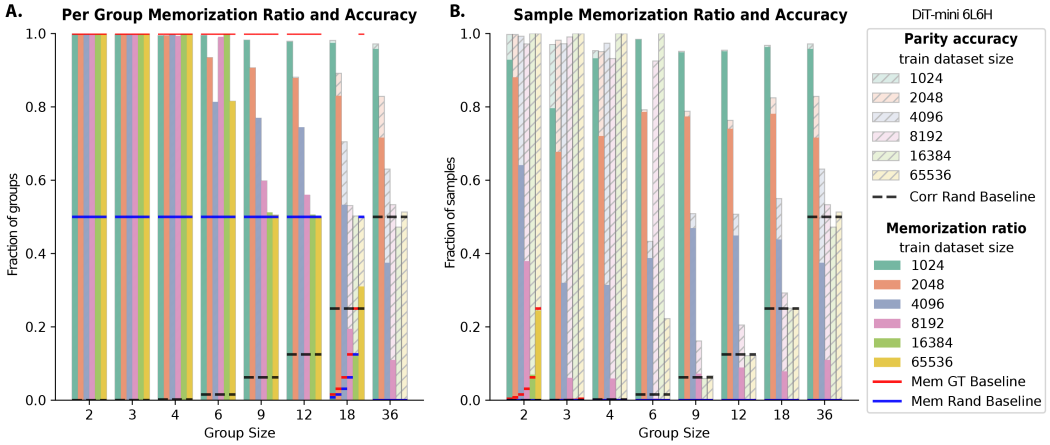


Figure 9: **Memorization and Creativity in Parity Learning across dataset scales.** Memorization ratio overlay on accuracy, for different group size and training dataset scale, at group level (**A.**) and sample level (**B.**), with similar format as Fig. 1**B,C**. Red solid line shows the memorization ratio of the ground truth distribution ($\mathcal{P}_G^+$ for groups, and $(\mathcal{P}_G^+)^{D/G}$ for samples); and blue solid line shows the memorization ratio of the chance distribution ($\mathcal{U}_G$ for groups and $\mathcal{U}_D$ for samples). Black dashed line shows the chance level accuracy.

C.2 Extended DiT learning curves

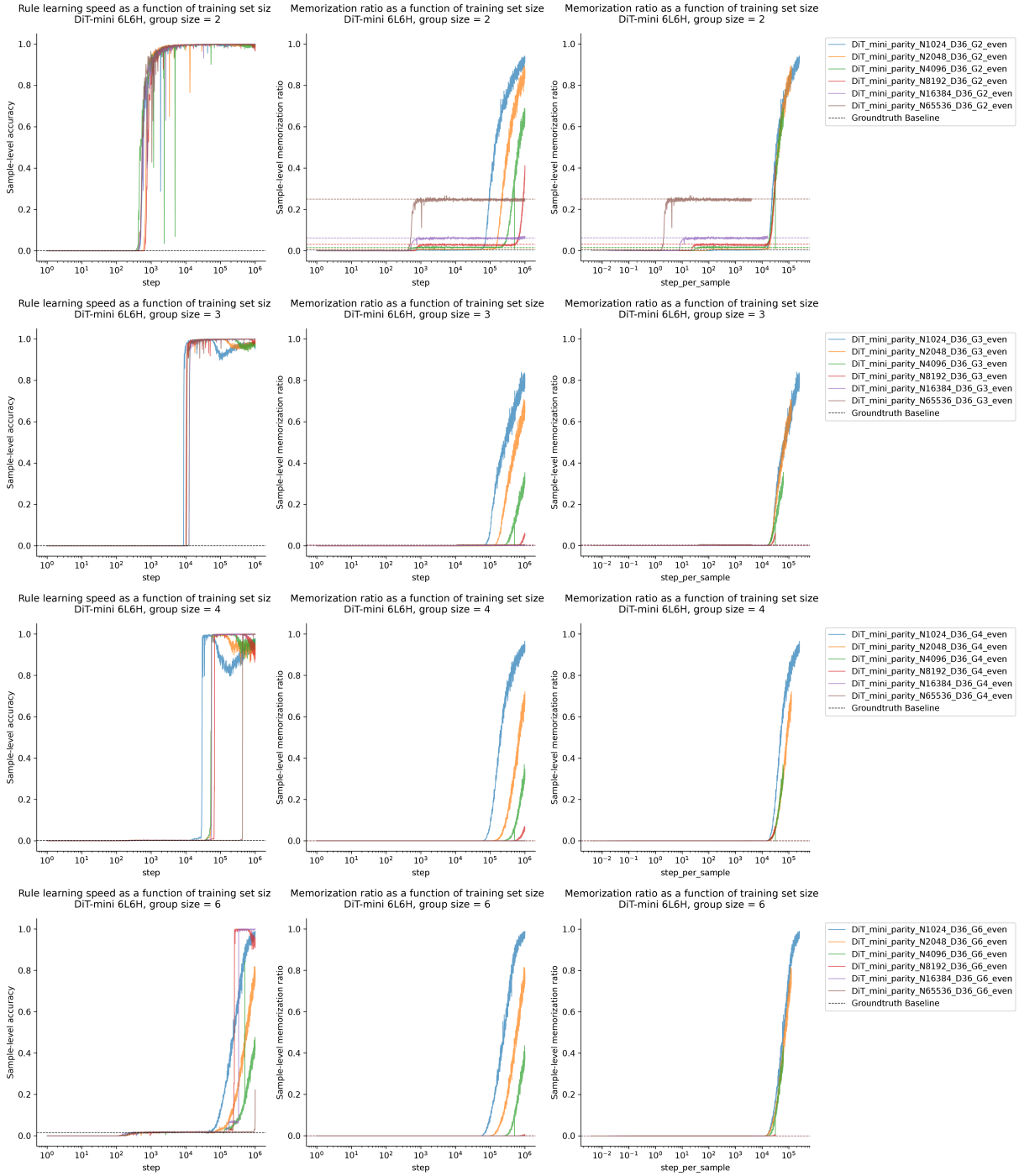


Figure 10: **Learning dynamics of rule acquisition and memorization across dataset size,** $G = 2, 3, 4, 6$**. Left.** Dynamics of sample parity accuracy across dataset scale, DiT-mini. **Mid. Right.** Dynamics of sample memorization ratio across dataset scales, the dynamics are plotted as a function of step (**Mid.**) and step per sample (step x batch size/ dataset size) (**Right.**). Colored dashed lines denotes the memorization ratio expected from the ground truth distribution. Per-G, per-N extension of the trajectories aggregated in Fig. 2**G,H**.

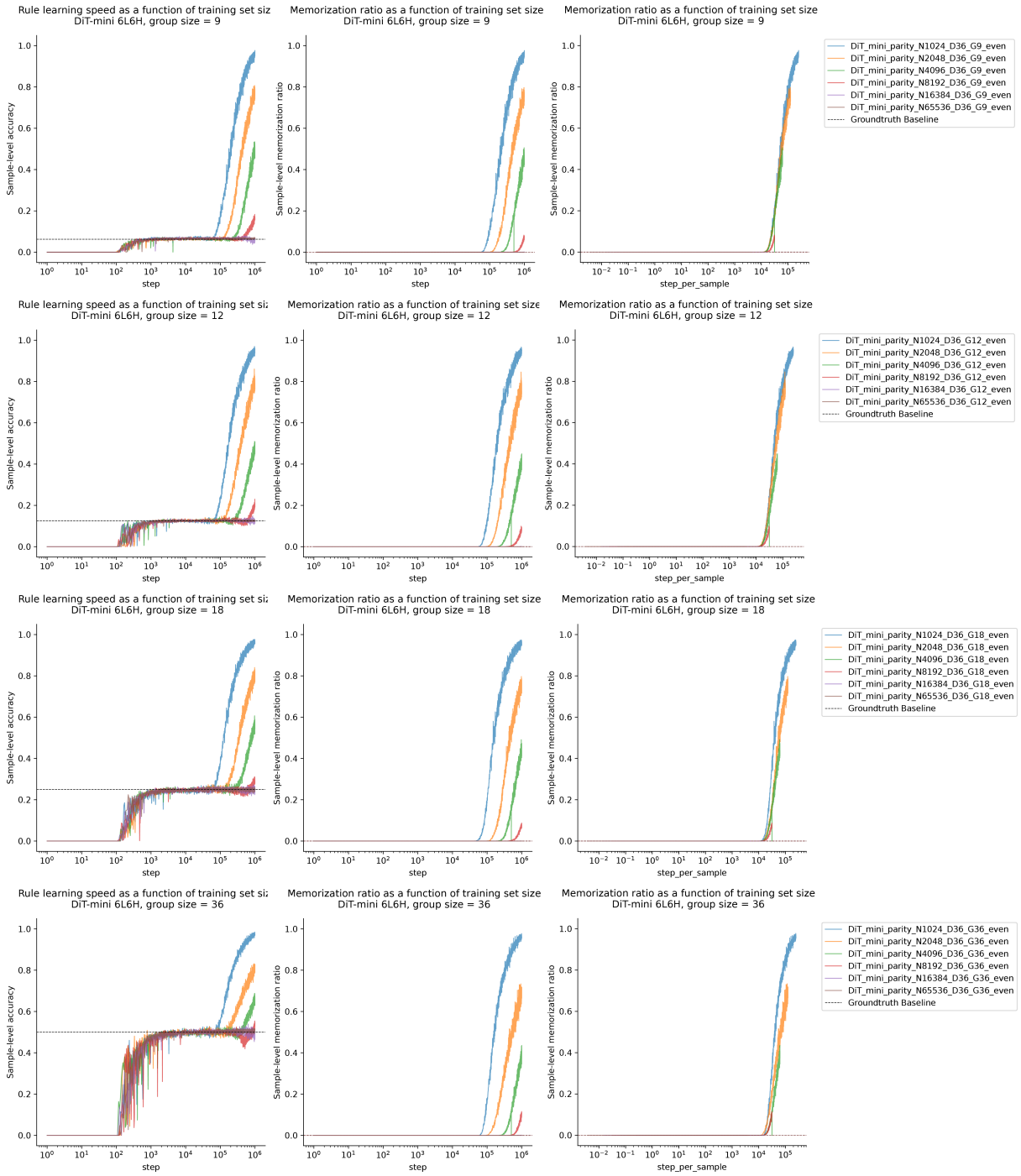


Figure 11: **Learning dynamics of rule acquisition and memorization across dataset size, $G = 9, 12, 18, 36$.** Similar format as Fig. 10; per-G, per-N extension of trajectories aggregated in Fig. 2**G,H**.

C.3 Extended memorization-time scaling for DiT and GPT

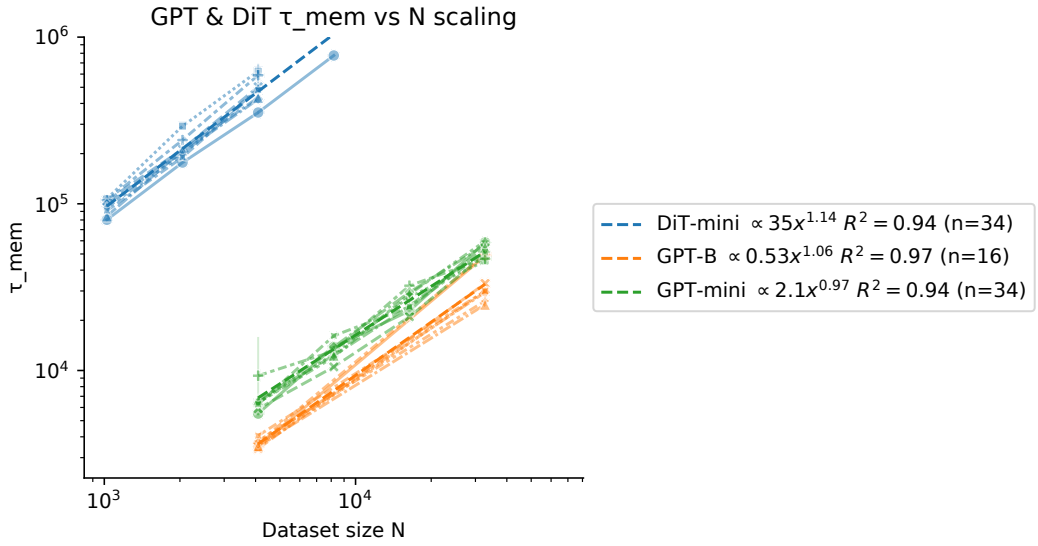


Figure 12: Memorization onset time τ_{mem} as a function of training set size N, for all DiT and GPT model scales and group sizes G. Each point is one experiment; dashed lines show power-law fits $\tau_{\text{mem}} \propto N^{\alpha}$ per model and G. The near-linear exponent $\alpha \approx 1$–1.1 is consistent across architectures and scales (see Table 5 for numerical values). DiT-mini exhibits systematically larger τ_{mem} than GPT-mini at matched N, resulting a systematic upward shift of scaling curve. This suggests that the denoising objective is harder to memorize relative to next-token prediction. GPT-B further shifts the scaling downward, suggesting larger model exhibits faster memorization.

Table 5 reports power-law fits $\tau_{\text{mem}} \approx c \cdot N^{\alpha}$ for each model and group size, using the adaptive memorization threshold $\theta_{\text{mem}} = 0.1 + N/\text{support_size}$ (see §B.3). Across models and all G values pooled, the exponent $\alpha \approx 1.0$–1.1, indicating a near-linear scaling of memorization time with dataset size.

Table 5: Power-law scaling of memorization onset τ_{mem} with training set size N: $\tau_{\text{mem}} \approx c \cdot N^{\alpha}$. Fit via log-log linear regression; R^2 and n denote fit quality and number of data points. τ_{mem} is measured with adaptive mem. threshold $= 0.1 + N/\text{support_size}$.

Model	G	c	α	R^2	n
DiT-mini	**all**	35.3	1.14	0.94	34
	2	45.8	1.08	1.00	5
	3	36.2	1.15	0.99	4
	4	19.0	1.25	0.99	4
	6	18.6	1.24	0.98	5
	9	70.2	1.05	1.00	4
	12	24.0	1.19	0.99	4
	18	26.4	1.17	1.00	4
	36	19.3	1.22	1.00	4
GPT-mini	**all**	2.1	0.97	0.94	34
	2	0.6	1.10	0.99	4
	3	1.1	1.02	0.99	4
	4	2.2	0.96	0.99	4
	6	11.0	0.81	0.83	6
	9	1.0	1.05	1.00	4
	12	0.7	1.09	1.00	4
	18	1.4	1.01	1.00	4
	36	2.2	0.97	0.98	4
GPT-B	**all**	0.5	1.06	0.97	16
	2	0.1	1.27	1.00	2
	3	0.4	1.10	1.00	2
	4	0.6	1.04	1.00	2
	6	0.1	1.25	1.00	2
	9	0.9	1.00	1.00	2
	12	1.7	0.93	1.00	2
	18	1.4	0.94	1.00	2
	36	1.4	0.96	1.00	2

C.4 Dissecting rule learning and memorization dynamics

C.4.1 Per-sample training dynamics

For sampling, we used a fixed set of 2048 initial Gaussian noise samples, and the deterministic Heun's sampler. Thus due to the continuous dependency of the sample on the PF-ODE parameters, individual sample trajectories are comparable over training time. At each checkpoint, each generated sample is assigned to one of four mutually exclusive states (see §B.3 for definitions): invalid (quantization-ambiguous), invalid (rule-error), valid & novel, and valid & memorized. This yields a per-sample state trajectory of shape $(T \times N)$ over the T saved checkpoints.

Transition matrix analysis. From consecutive checkpoint pairs $(t, t{+}1)$ we compute a $4{\times}4$ transition count tensor $\mathbf{T}(t)$, where $\mathbf{T}[t, i, j]$ is the number of samples in state i at step t that move to state j at step $t{+}1$. Row-normalizing gives the empirical transition probability matrix. To characterize different training phases, we aggregate $\mathbf{T}(t)$ over key time windows—pre-rule-learning, rule-learning, and memorization—by summing counts within each window (Fig. 13). We also animate the heatmap over the full training trajectory to reveal how transition dynamics evolve continuously.

One salient observation from Fig. 13 is that, at the late memorization phase, looking at the transition probability matrix (Markov matrix), we can see invalid (rule-error) is *not an attractor state*, i.e. samples entering it will jump to valid & novel, and valid & memorized in the next model state. In contrast, valid & novel and valid & memorized are both attractor states, with some transition probability between them. We note that the valid $\rightarrow$ memorized transition slightly outnumbered the reverse direction causing the raw increase of memorized samples. This suggests the picture that the memorized samples are attractor states and tend to stay memorized, while the rule violation is transient state, and likely some "side effect" of the landscape change.

Hamming distance conditioned on sample type. Beyond the four states, for each sample, we also record its Hamming distance to the nearest training-set member. We then condition this statistic on specific sample types, (i) all samples, (ii) all rule valid samples, (iii) rule valid and novel samples (i.e. excluding hamming 0 case); and then track statistics of these distance across training time. Across all G rules, we find that the distance to training set of all three types decay at the late phase of training ($[1E5, 1E6]$); and this decay is evident even for valid and novel samples, showing that even before exact memorization, the generated samples get closer to the training point cloud in the late phase. Consistently, this decay of hamming distance precedes the onset of memorization τ_{mem} measured from sample level (Fig. 14). Thus it shows that at the late phase, the training points have increasingly strong "attraction force", pulling the novel generations towards the ones closer to training data.

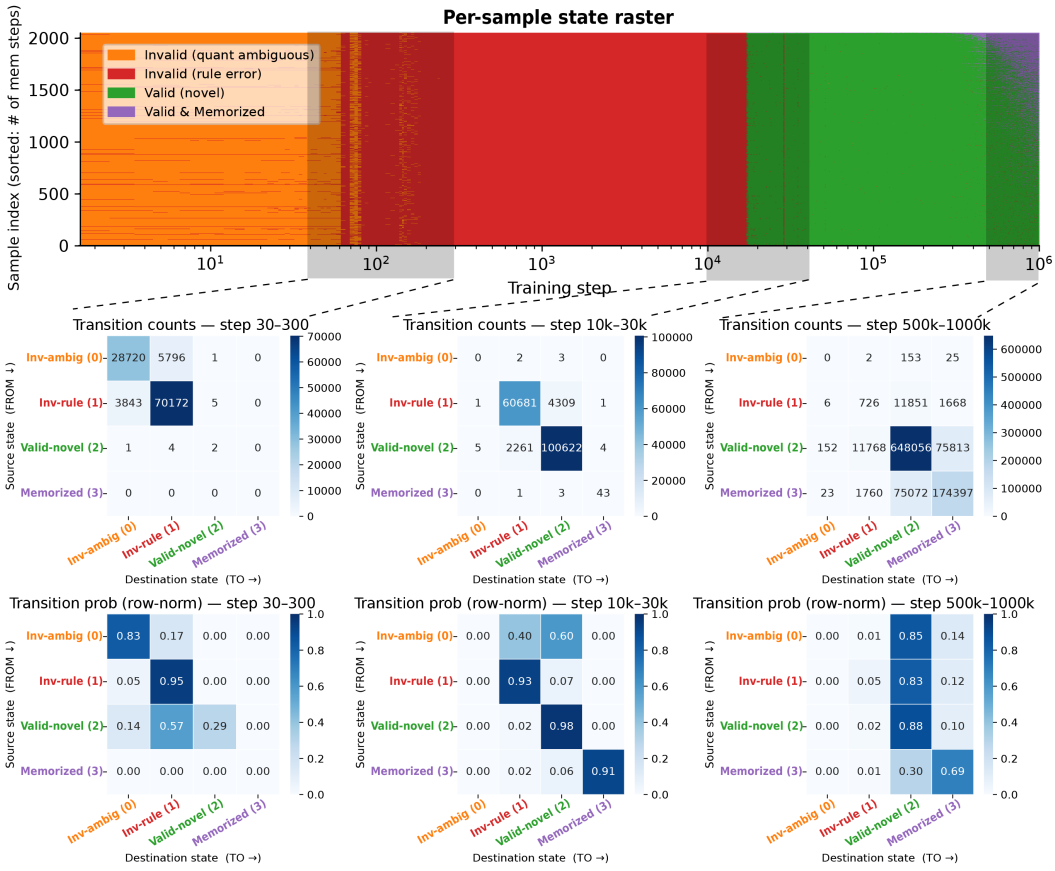


Figure 13: **Per-sample state transitions reveal staged diffusion learning. Top**: raster plot tracking the state of each generated sample across training, sorted by the step at which it first becomes memorized, same panel as Fig. 3**A**. **Middle**: transition count matrices between sample states in three representative training windows: early quantization transition (steps 30–300), rule-learning transition (steps 10k–30k), and late memorization onset (steps 500k–1000k). Rows denote source states and columns denote destination states between consecutive evaluation checkpoints. **Bottom**: row-normalized transition probabilities for the same windows. Early in training, samples primarily move from quantization-invalid to rule-invalid states; during rule learning, rule-invalid samples transition into the valid-novel state; late in training, valid-novel samples increasingly transition into the memorized state. Together, these dynamics show that memorization is preceded by a broad valid-but-novel phase rather than emerging directly from invalid generations.

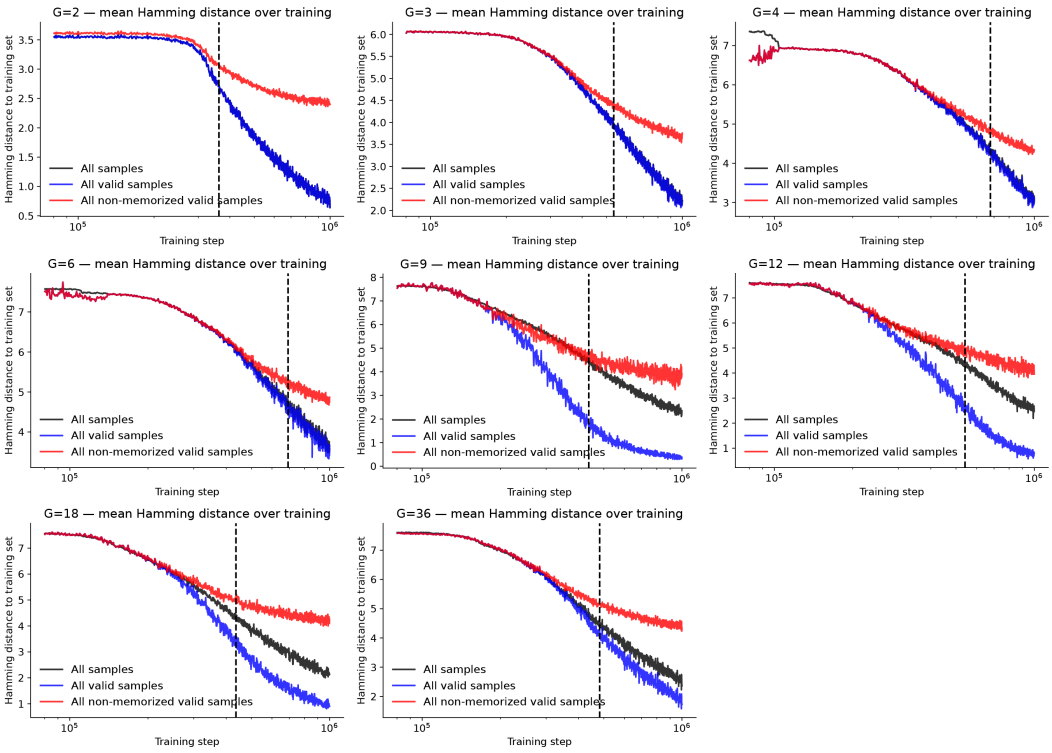


Figure 14: **Valid generations move progressively closer to the training set before memorization.** Mean Hamming distance from generated samples to the closest training set sample over late training, shown separately for each parity group size G. Black curves show the average over all generated samples; blue curves show the average restricted to rule-valid samples; red curves show rule-valid but non-memorized samples. Dashed vertical lines mark the memorization onset τ_{mem} with adaptive threshold memorization ratio 0.1.

As training proceeds, the nearest-neighbor Hamming distance of valid samples decreases sharply and continues to decline around τ_{mem}. Importantly, the red curves also decrease, showing that even non-memorized valid samples become increasingly close to training samples before exact memorization. Thus, the late training phase is not an abrupt jump from novelty to memorization, but a gradual contraction of valid generations toward the empirical training set.

C.4.2 DSM loss by noise scale

We evaluate the DSM loss on three held-out data splits at each checkpoint:

1. **Training set**: the $N{=}4096$ samples used during training.
2. **Valid-novel test set**: $N{=}4096$ rule-valid samples drawn from the same group-parity distribution but verified to be non overlapping with training sample (i.e. not memorizable).
3. **Boolean-cube samples**: $N{=}4096$ uniformly random $\{\pm 1\}^D$ binary vectors with no parity constraint.

For model checkpoint we compute the unweighted DSM loss with respect to each split and evaluate over a fixed grid of noise levels $\sigma \in [\sigma_{min}, \sigma_{max}] = [0.002, 80.0]$ (log-spaced, 50 levels), giving a noise-scale-vs-step matrix per data split.

We visualize these matrices through several slices. We plot the full loss-vs-σ spectrum (log–log scale) at selected checkpoints to reveal *at which noise scales* the three splits diverge (Fig. 3**C**, 15). These spectral view identifies the noise window that is critical for rule learning ($[0.1, 2.0]$) and memorization ($[0.2, 2.0]$).

At these critical noise scales, we also plot the three DSM loss traces across training time to observe the onset of the splits (Fig. 3**B**, 16). The phase transitions identified from samples can also be identified from the loss: The rule-learning transition is identified as the step where the boolean-cube loss diverges from the train and valid-novel curves; the memorization onset is identified as the step where the training and valid-novel curves diverge from each other. Notably, the DSM loss splits happen before the corresponding τ_{rule} and τ_{mem} identified from samples.

Finally, visualizing the train-test difference of the DSM loss (i.e. generalization gap), we can trace the onset of the split on the noise-level $\times$ training step plane (Fig. 17). One can see the splits first happens in $\sigma \sim [0.5, 1.0]$ then propagates to the lower and higher noise levels. We hypothesize that the attractor landscape first changes at $\sigma \sim 1.0$, then due to the shared parametrization of denoiser, the change propagates to landscapes at nearby noise scales, causing the enlargement of the basins at $\sigma \sim 0.5$.

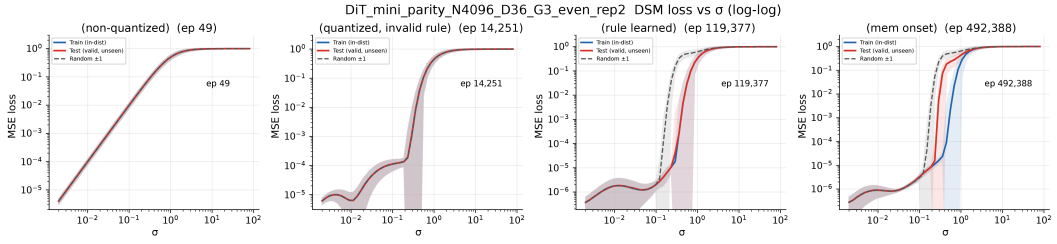


Figure 15: **Noise-scale spectra at representative training phases.** DSM loss as a function of σ is shown at the four checkpoints: early non-quantized generation, quantized but rule-violating generation, rule-learned generation, and memorization onset (same as Fig. 3**D**). Train, held-out valid, and random Boolean-cube samples are nearly indistinguishable before rule learning. After rule learning, the random Boolean-cube spectrum separates from train and held-out valid loss spectrum; at memorization onset, the training spectrum separates from the held-out valid spectrum. Both separations are most pronounced at only a narrow range of noise scales $[0.1, 2.0]$ for rule learning, $[0.2, 2.0]$ for memorization, supporting the view that rule learning and memorization correspond to vector-field changes at a specific spatial and noising scale.

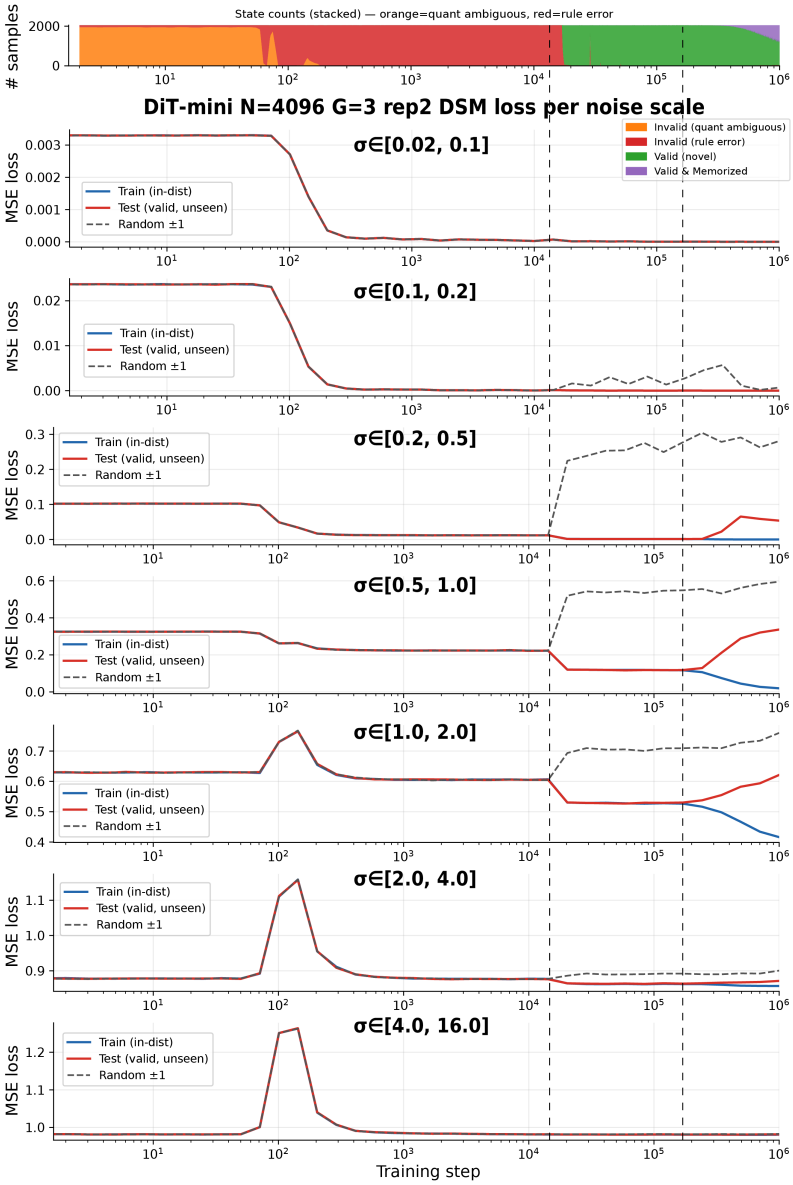


Figure 16: **DSM loss dynamics across noise-scale bands.** For DiT-mini trained on parity with $G{=}3$, $N{=}4096$ (run 2), we track DSM losses for training samples, held-out rule-valid samples, and random Boolean-cube samples, separately across logarithmic noise-scale bands. The top panel shows the corresponding sample-state counts across training (Fig. 3**A**). Low-noise losses quickly collapse across splits as the model learns the Boolean cube, while intermediate noise scales show the parity-task relevant separations: random Boolean-cube loss separates from train/test around rule learning, and train/test losses separate around memorization. Vertical dashed lines mark train-random and train-test loss splitting times. Note that the DSM loss split times preceeds the sample level rule learning and memorization which is used to define τ_{rule} and τ_{mem} in the main paper. This makes sense, since the change of vector field is immediately reflected in the DSM loss, and the samples are integrated result of vector field, thus the change is evident there later. In the future work, one could define τ_{rule} and τ_{mem} according to DSM loss splits.

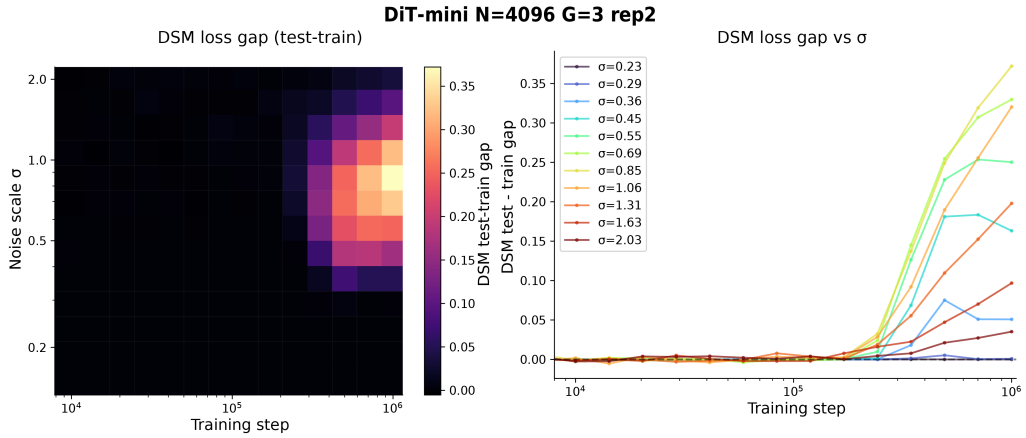


Figure 17: **Onset of Train–test DSM loss gap zooming in the critical noise scales** $[0.2, 2.0]$**. Left**: heatmap of the DSM loss gap between held-out rule-valid samples and training samples as a function of training step and noise scale σ, focusing on the critical noise scale. **Right**: the same gap plotted over training for representative fixed σ values. We note that the train-test gap first opens for noise scales around $\sigma \in [0.5, 1.0]$, then this gap propagates and affects the lower and higher noise scales. In the end, the gap in $\sigma \in [0.5, 1.0]$ is still the most pronounced.

C.4.3 Vector-field and attractor-basin analyses

Plane definition. All vector-field and landscape visualizations are performed in a 2D affine subspace of $\mathbb{R}^D$ anchored at a training sample $\mathbf{x}_a$. Two companion anchor points define the spanning vectors: $\mathbf{x}_b$ is obtained from $\mathbf{x}_a$ by flipping bits 0 and 1 simultaneously (Hamming-2 flip, producing a *valid-novel* neighbor), and $\mathbf{x}_c$ is obtained by flipping bit 0 alone (Hamming-1 flip, producing an *invalid* neighbor). The two plane vectors are $\mathbf{v}_{ab} = \mathbf{x}_b - \mathbf{x}_a$ and $\mathbf{v}_{ac} = \mathbf{x}_c - \mathbf{x}_a$, Gram–Schmidt orthonormalized. A 50×50 grid spanning $\alpha, \beta \in [-1.75,\ 3.75]$ is constructed as $\mathbf{x}(\alpha, \beta) = \mathbf{x}_a + \alpha\,\mathbf{v}_{ab} + \beta\,\mathbf{v}_{ac}$, and the neural network denoiser $D_\theta(\mathbf{x}, \sigma)$ is evaluated at each grid point at representative noise scales σ, then we can derive the score via Tweedie's formula $D_\theta(\mathbf{x}, \sigma) = \mathbf{x} + \sigma^2 s_\theta(\mathbf{x}, \sigma)$.

We note that these four points form a face of the boolean hypercube, and their coordinates under the normal element-wise basis should be $(-1,-1), (-1,1), (1,-1), (1,1)$. The planes showed in Fig. 3**D** uses x_a, x_b line as the x-axis, thus the coordinate system is rotated and shifted, but not scaled. In this 2d plan, we know the exact coordinates of the four anchors $x_a = (0,0); x_b = (2\sqrt{2}, 0), x_c = (\sqrt{2}, \sqrt{2}), x_d = (\sqrt{2}, -\sqrt{2})$. This geometry also defines the critical distance of our parity datasets: the distance to nearest rule invalid sample is 2 (Hamming 1); the distance to the nearest valid and novel sample is (usually) $2\sqrt{2}$ (Hamming 2).

Visualization. Each checkpoint at each noise scale σ is displayed as a column of two rows:

Top row — score magnitude and displacement field. The heatmap shows the ℓ_2 norm of the full-dimensional score vector $s_\theta(\mathbf{x}, \sigma)$ at each grid point, rendered in the `magma` colormap (bright = high score magnitude). Thus the darker color, i.e. close to zero score magnitude suggests attractor of the dynamics $D_\theta(\mathbf{x}, \sigma) = \mathbf{x}$. Overlaid white arrows show the 2D projection of the score vector $s_\theta(\mathbf{x}, \sigma)$ onto the $(\mathbf{v}_{ab}, \mathbf{v}_{ac})$ plane, subsampled for clarity. Colored markers indicate the four anchor points: $\mathbf{x}_a$ (white, training sample), $\mathbf{x}_b$ (cyan, valid-novel Hamming-2 neighbor), $\mathbf{x}_c$ (yellow, invalid Hamming-1 neighbor), and $\mathbf{x}_d$ (orange, the mirror of $\mathbf{x}_c$ across $\mathbf{x}_a$).

Bottom row — signed attraction along the $\mathbf{v}_{ab}$ axis. The heatmap shows the scalar projection of the denoiser onto $\mathbf{v}_{ab}$, i.e. $D_\theta(\mathbf{x}, \sigma) \cdot \mathbf{v}_{ab}$, in a diverging `RdBu_r` colormap. Due to the specific geometry of the four points on the face of hypercube, we can read out the target of attraction easily from the color: blue = attracted toward $\mathbf{x}_a$; red = attracted towards $\mathbf{x}_b$; white = attracted towards $\mathbf{x}_c$ or $\mathbf{x}_d$. This row directly visualizes the basin and the boundary separating the $\mathbf{x}_a$ attractor from the $\mathbf{x}_b, \mathbf{x}_c, \mathbf{x}_d$ attractors.

The same plane, grid, and anchor points are reused identically across all checkpoints (step $\approx$ 50, 14k, 119k, 492k for Fig. 3**D**); grid evaluations are cached by a hash of the plane geometry and σ value to avoid redundant forward passes. A more densely sampled landscape view across the training process is shown in Figs. 19–21.

Attractor basin profile quantification. To quantify the spatial extent of the attractor basin around a training sample $\mathbf{x}_a$, we evaluate the denoiser along 1D line segments through $\mathbf{x}_a$ in three fixed directions: (i) toward a Hamming-1 invalid neighbor $\mathbf{x}_{\text{inv}}$ (one bit flipped, breaking group parity); (ii) toward a Hamming-2 valid-novel neighbor $\mathbf{x}_{\text{nov}}$ (two bits within the same group flipped, parity preserved, not in the training set); (iii) toward the nearest other training sample $\mathbf{x}_{\text{train}}$ (by Hamming distance). Each line is parameterized as $\mathbf{x}(t) = \mathbf{x}_a + t\,(\mathbf{x}_{\text{end}} - \mathbf{x}_a)$, with t ranging from -0.5 (behind $\mathbf{x}_a$) to 2.0 (past the endpoint), sampled at n=150 uniformly spaced values. At each point $\mathbf{x}(t)$ the denoiser output $D_\theta(\mathbf{x}(t), \sigma)$ is computed. Three basin metrics are derived:

1. **Exact match**: $\mathbf{1}[\text{sign}(D_\theta(\mathbf{x}(t))) = \text{sign}(\mathbf{x}_a)]$ — binary indicator that all output bits (after binarization) agree with $\mathbf{x}_a$; collapses to 0 when the denoiser is attracted to a different vertex.
2. **Hamming distance**: $\sum_i \mathbf{1}[\text{sign}(D_{\theta,i}) \neq \text{sign}(x_{a,i})]$ — number of bits that disagree, a graded measure of basin exit.
3. **L2 distance from start**: $\|D_\theta(\mathbf{x}(t)) - \mathbf{x}_a\|_2$ — Euclidean distance of the denoised output from the basin center.

The procedure is repeated independently for N_{samp}=30 different anchor samples $\mathbf{x}_a$ drawn from the training set. All three metrics are averaged across anchors, and bootstrap confidence intervals (5th–95th percentile of the bootstrap distribution of the mean, with 2000 resamples) are shown as

shaded bands. Results are reported at σ=0.5 for three representative checkpoints: ep 14k (rule-error phase), ep 119k (generalization phase), and ep 492k (memorization phase), in Fig. 18.

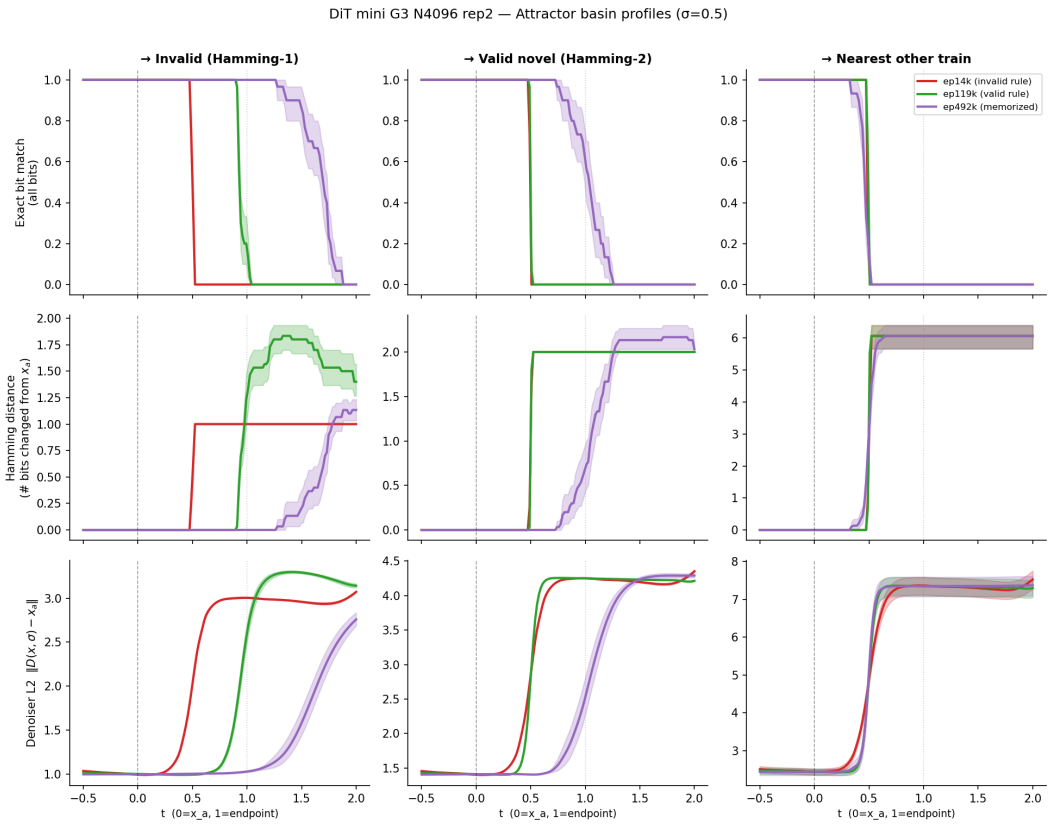


Figure 18: **Attractor-basin profiles across training phases.** For a DiT-mini model trained on group parity (G=3, N=4096), we probe 1D line segments from a training sample x_a toward three endpoints: an invalid Hamming-1 neighbor (left), a valid-novel Hamming-2 neighbor (middle), and the nearest other training sample (right). Rows show exact-bit match to x_a, Hamming distance from x_a, and denoiser ℓ_2 distance $\|D(x,\sigma) - x_a\|$ at σ=0.5. The basin around x_a expands from the rule-error phase (14k) to the rule-learned phase (119k), and expands further at memorization onset (492k), especially toward valid-novel neighbors.

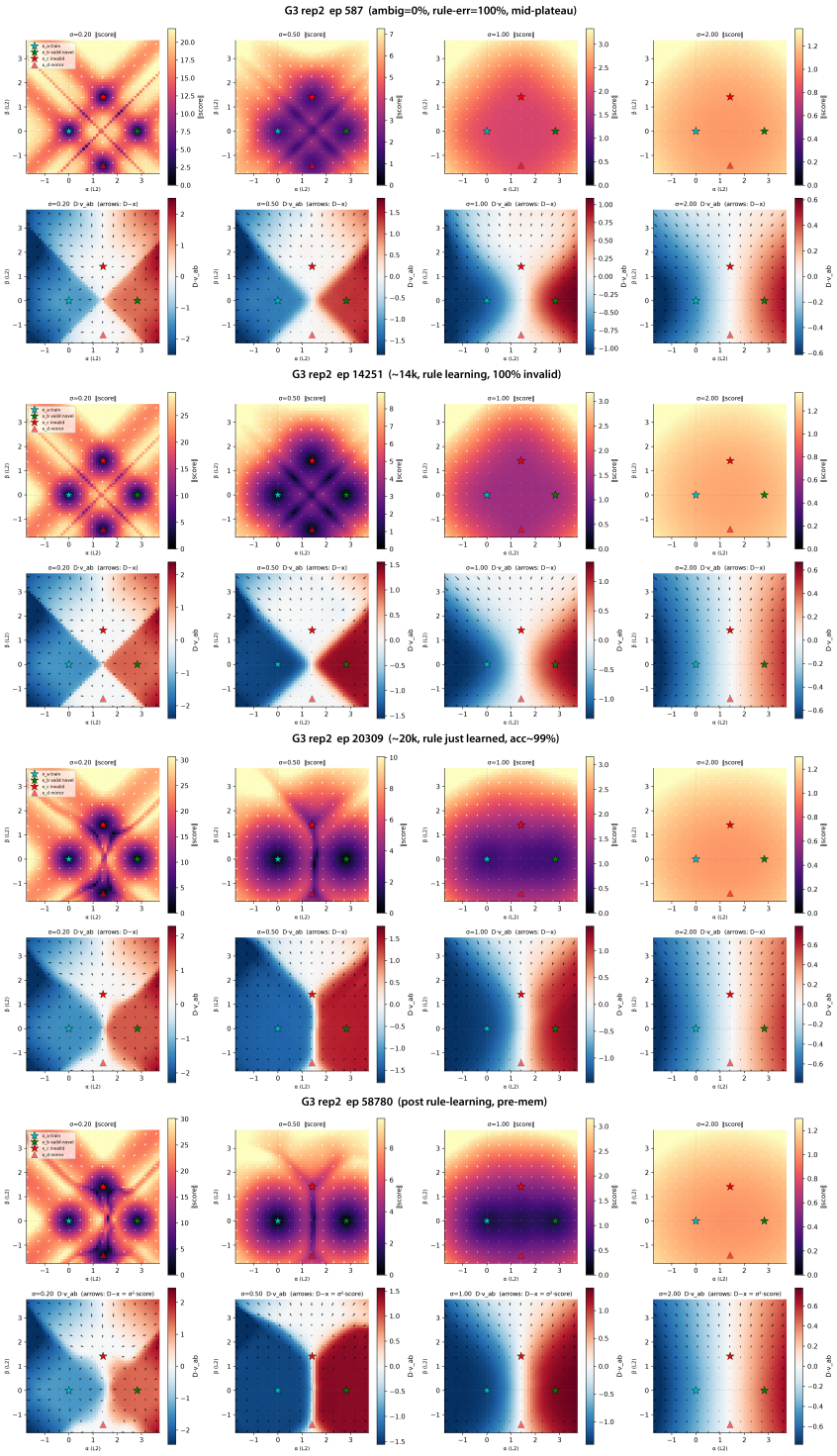


Figure 19: **Early vector-field landscape evolution.** Score magnitude (top row of each checkpoint) and signed denoising displacement along $\mathbf{v}_{ab}$ (bottom row) are shown for $\sigma \in \{0.20, 0.50, 1.00, 2.00\}$. Before rule learning, the landscape first forms attractors around Boolean-cube vertices; after rule learning begins, the valid anchors $\mathbf{x}_a$ and $\mathbf{x}_b$ start to dominate the rule-violating anchors $\mathbf{x}_c$ and $\mathbf{x}_d$.

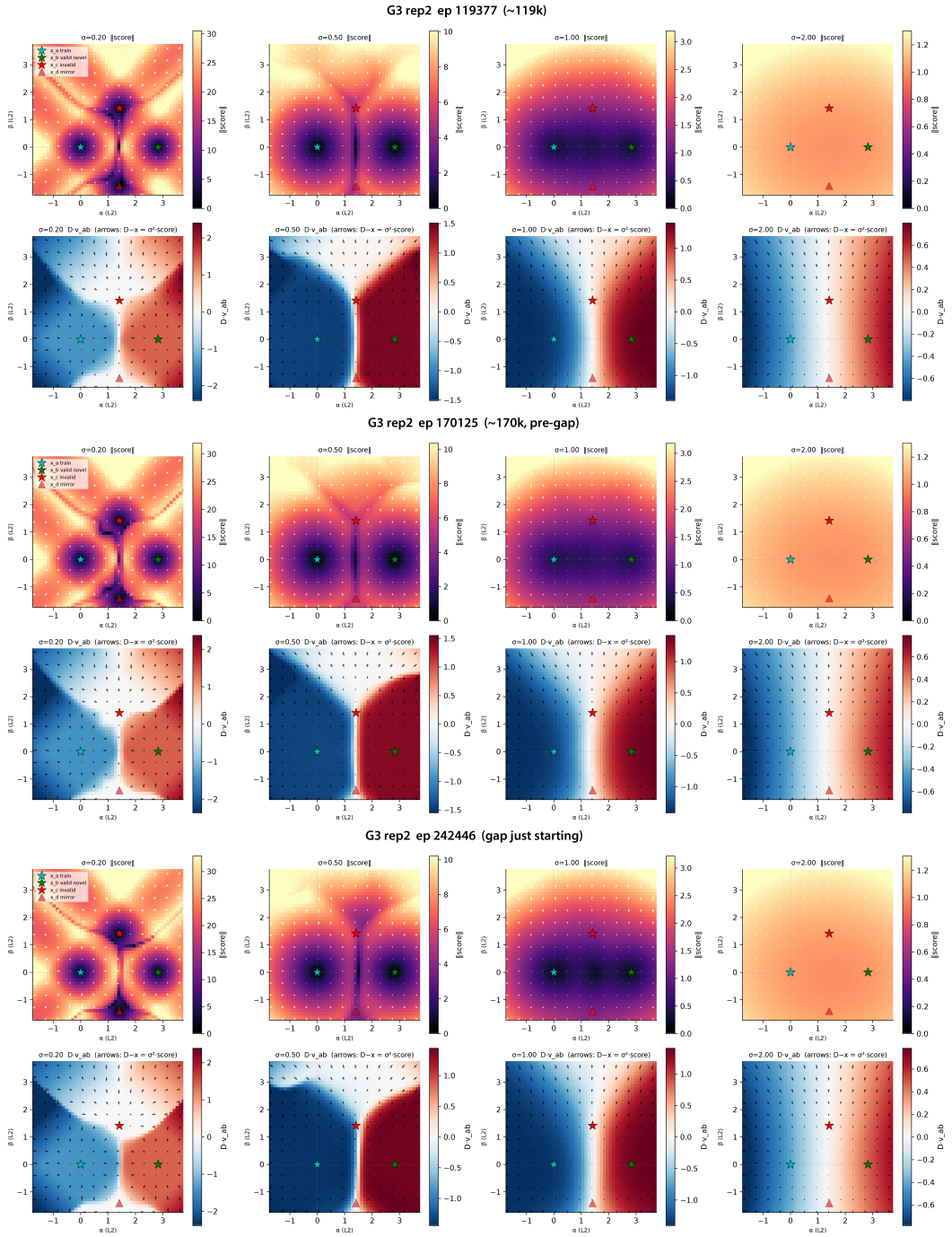


Figure 20: **Mid-training vector-field landscape evolution.** The same 2D slice and noise scales are tracked through the generalizing phase. The basin boundary between the training sample $\mathbf{x}_a$ and valid-novel neighbor $\mathbf{x}_b$ remains visible while the attraction toward $\mathbf{x}_a$ progressively strengthens, especially at intermediate noise scales.

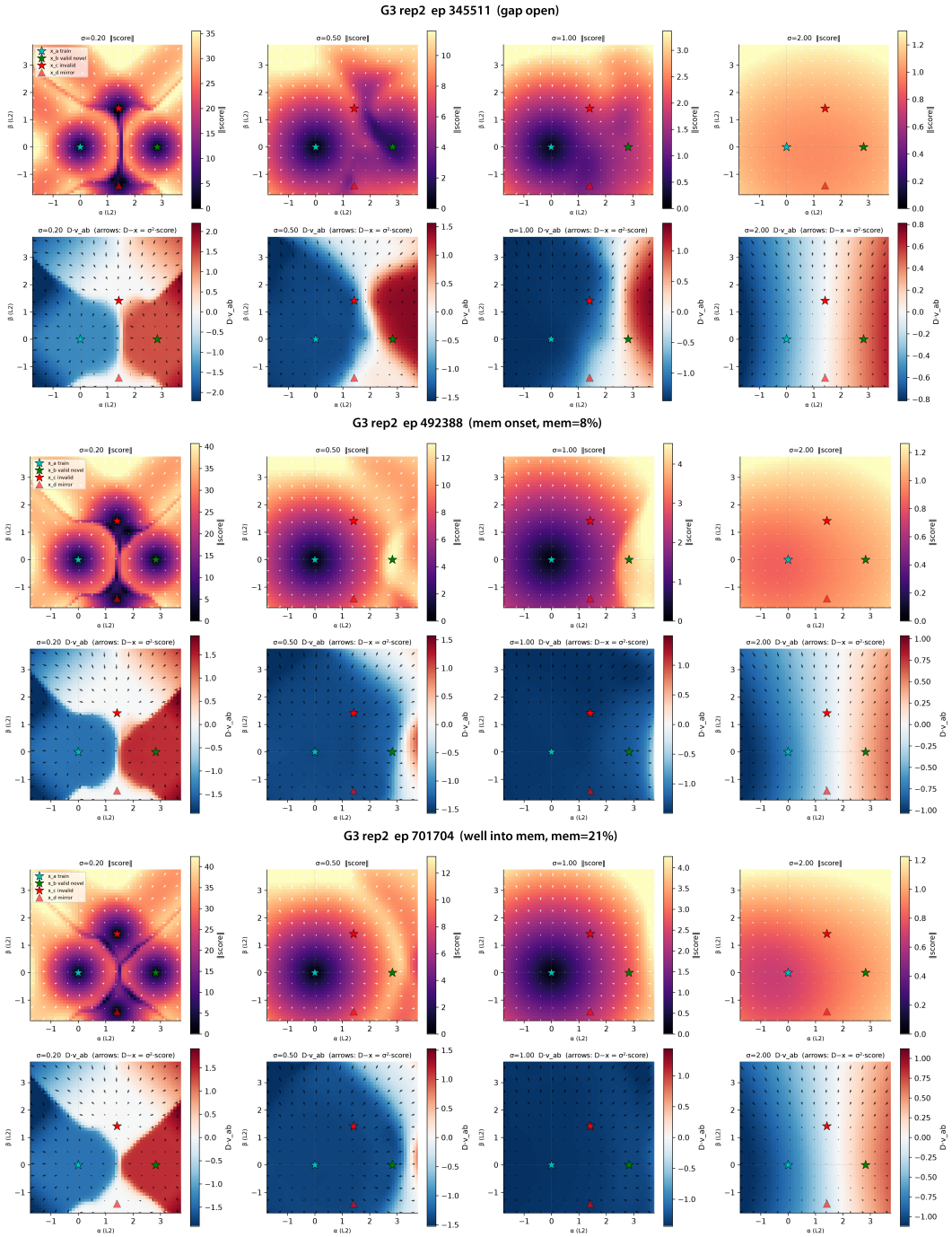


Figure 21: **Late vector-field landscape evolution.** During memorization onset and late memorization, the attraction basin of the training sample $\mathbf{x}_a$ expands toward $\mathbf{x}_b$ and eventually dominates much of the plane. The effect is strongest around the same intermediate noise scales where the DSM train–test gap opens.

C.5 Autoregressive vs diffusion supplementary results

Generation paradigm. GPT generates bits sequentially in a fixed left-to-right order, while DiT generates all D bits jointly in a single diffusion process (conditioned on noise level σ). This fundamental difference leads to distinct inductive biases: GPT can exploit causal structure (later bits can be predicted from earlier ones), whereas DiT treats all bit positions symmetrically. In accordance with this difference, the GPT models employ causal attention masks, while DiT use all-to-all self attention.

Structural consequence for parity learning. For even-parity groups of size G, the last bit of each group is fully determined by the preceding $G-1$ bits. In the GPT, this determinism is directly exploited during next-token prediction: once the model learns the parity rule, the CE at positions $G, 2G, \ldots, D$ collapses to zero first, before other positions. The DiT, by contrast, has no privileged ordering and learns to jointly enforce the constraint across all positions simultaneously, making its rule learning detectable as a collective change in the score field at a specific noise level σ rather than a position-specific collapse.

Shared qualitative dynamics. Despite these architectural differences, both models exhibit the same two-phase dynamics: (i) an initial rule-learning transition in which validation-set performance improves sharply, and (ii) a subsequent memorization transition in which training-set performance diverges from held-out performance. The timescales differ: GPT-mini reaches both transitions within 10^5 gradient steps, while DiT-mini requires up to 10^6 steps due to the harder denoising objective and the continuous-noise parameterization. Both models are compared under identical data conditions: N=4096, D=36, G=6.

Figure 22 shows the GPT analogue of Figure 2, confirming that the two-clock phenomenology holds for autoregressive models.

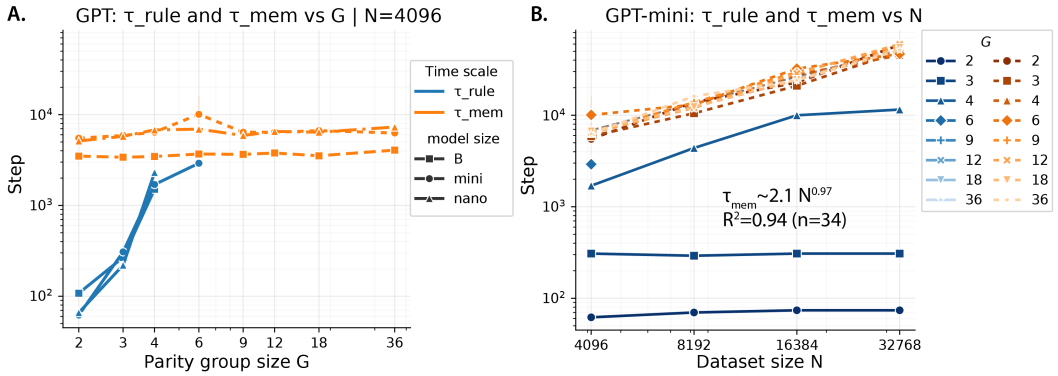


Figure 22: **Two clocks τ_{rule} and τ_{mem} scaling relationships for GPT models — analogous layout to Fig. 2. A.** τ_{rule} (blue) and τ_{mem} (orange dashed) vs. G at N=4096 for GPT-nano, mini, and B; τ_{rule} shown only for learnable G where τ_{rule} is smaller than τ_{mem}. **B.** (i) τ_{rule} vs. N for learnable G. (ii) τ_{mem} vs. N across all G, with power-law fit $\tau_{\text{mem}} \approx 2.1\, N^{0.97}$ (R^2=0.94, n=34, GPT-mini).

C.6 Beyond Parity Rules

τ_{rule} and τ_{mem} beyond parity rules: exact-K and row-K For these alternative rules, we found similar two clock structure in their learning dynamics, a time point where rule gets learned and a time point where memorization starts. In general, the count based rules are much faster to learn than the parity rules (Fig. 23**A**).

For simple count based rules, like exactK and rowK, the rule learning time τ_{rule} is similar to the simplest parity rule $G = 2$, saturating in 10^3 steps. Increasing the number of requird active bits K in exact-K rule doesn't quite modulate the τ_{rule} (Fig. 23**B**). This is stark contrast in changing G for parity rules, which significantly delays the rule learning. Notably row-K, which is group level K bits active rule, doesn't delay the learning, but make τ_{rule} even smaller, potentially because fewer bits interact (Fig. 23**C**). These results support the energy-based interpretation above: exact-K and row-K constraints can be represented by degree-2 penalty on sum of weights, so their rule-learning clocks are much earlier than high-degree group parity. Rules with more complex K structure (i.e. allowing K to be chosen from a list) slightly delays τ_{rule}, but the general structure are consitent with

Consistent with our main result, the memorization time scale for alternative rules is relatively insensitive to the specific rule and remains largely set by dataset size and model capacity (Fig. 23**A**). The resulting gap between early rule acquisition and late training-set fit produces an innovation window across these non-parity binary tasks.

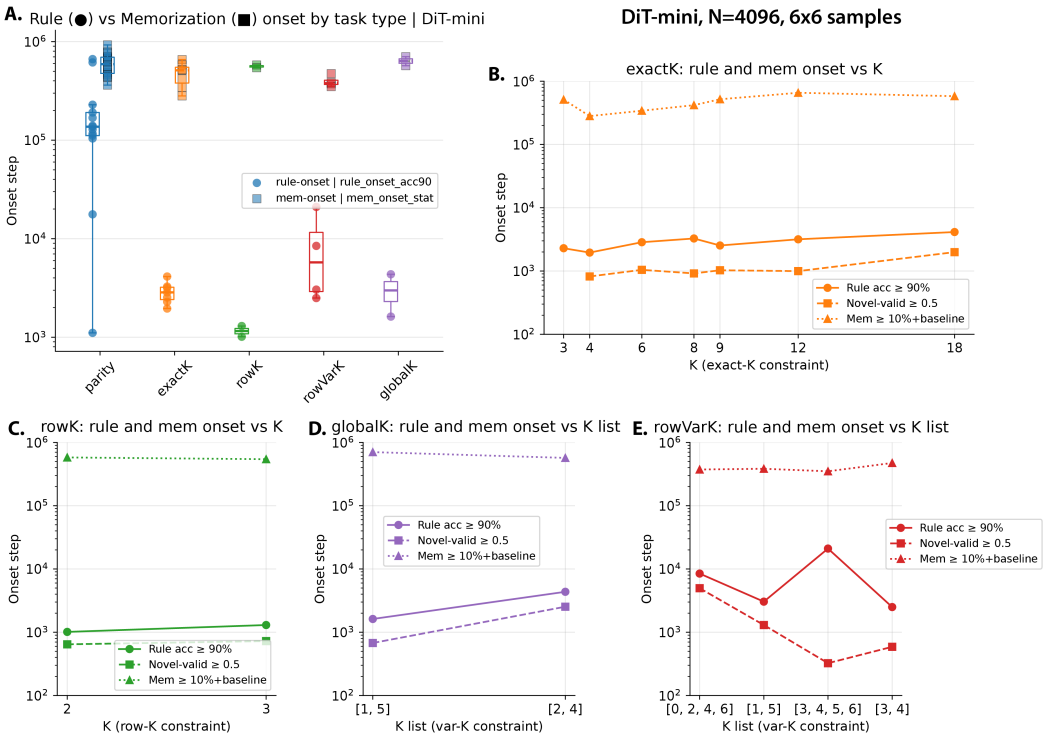


Figure 23: **Rule-learning and memorization timescales beyond parity.** DiT-mini models are trained on 6×6 binary samples with N=4096. **A.** Rule onset (circles; sample rule accuracy $\geq 90\%$) and memorization onset (squares; memorization ratio above baseline) across parity, exact-K, row-K, row-variable-K, and global-K tasks. **B–E.** Onset times across different K values or K lists for exact-K, row-K, global-K, and row-variable-K tasks, with novel-valid onset shown when available. Counting-style constraints are learned in 10^3–10^4 steps, substantially earlier than their memorization onsets, while memorization remains concentrated near late training across task variants.

Beyond parity rules and binary samples: SuDoKu and Latin Square Generalizing beyond rules applying to binary valued samples (boolean cube), we also examined more naturalistic rules such as Sudoku and Latin Square. These structured categorical tasks reveal a hierarchy of rule-learning

difficulty that is qualitatively different from group parity. In the parity task, increasing G makes the local rule itself higher-order and eventually prevents rule acquisition, causing late performance to become memorization-driven. Here, by contrast, the primitive constraints are simpler categorical uniqueness rules, but the difficulty arises from composing them across spatial axes and blocks.

RowOnly is learned rapidly, as it only requires satisfying row-wise constraints. The τ_{rule} is comparable to easier parity rules. Latin Square and Sudoku are delayed because the model must coordinate row and column constraints, and the column-valid ratio rises much later than the row-valid ratio. This suggests an anisotropic learning order: DiT first captures the more local or easier row structure before learning cross-row column consistency. Sudoku, which adds block constraints, reaches high full-sample validity in this setting, but also shows a stronger late increase in sample memorization, especially in the scalar setting.

Thus, unlike high-G parity where failure reflects difficulty of learning the rule, structured categorical tasks expose a different bottleneck: individually learnable rules are nested to constrain the globally consistent samples.

One notable finding is that, comparing between encoding schemes, when one-hot encoding is used for these rules, the memorization ratio is much smaller, or i.e. memorization is delayed beyond this time horizon. This suggests the τ_{mem} depends also on the encoding of the input, i.e. the geometry of the samples in the space. This is consistent with the view that the basin needs to expand enough to devour nearby valid samples to cause increase of memorized samples. One hot encoding have a higher dimensional raw input space and likely larger distance between valid samples, thus memorization onset is later.

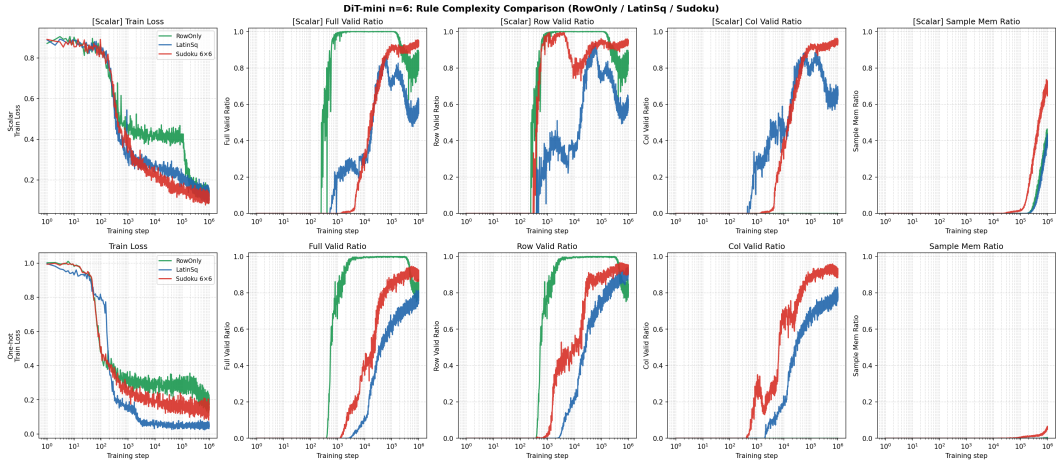


Figure 24: **Learning dynamics for structured categorical rule families: row only (Permutation), LatinSquare, Sudoku.** Training trajectories of DiT-mini on three $n = 6$ structured categorical datasets: RowOnly (row permutation), Latin Square, and 6×6 Sudoku with 2×3 blocks. Columns show train loss, full-sample valid ratio, row-valid ratio, column-valid ratio, and sample memorization ratio over training. The top row shows training results with scalar encoding (input shape $1 \times 6 \times 6$), while the bottom row shows training results with one-hot encoding (input shape $6 \times 6 \times 6$). RowOnly requires only row-wise constraints; Latin Square requires both row and column constraints; Sudoku adds block constraints on top of row and column constraints. Across tasks, RowOnly is learned earliest, reaching high row and full-sample validity around 10^3 steps, and the rule learning transition is sharp, similar to parity learning. However, Latin Square and Sudoku exhibit delayed and more gradual validity increase, with column-level validity emerging later than row-level validity. Sudoku reaches high full-sample validity more reliably than Latin Square in this run, but also shows the strongest late memorization. Notably, one-hot training generally shows much lower memorization over the same horizon.

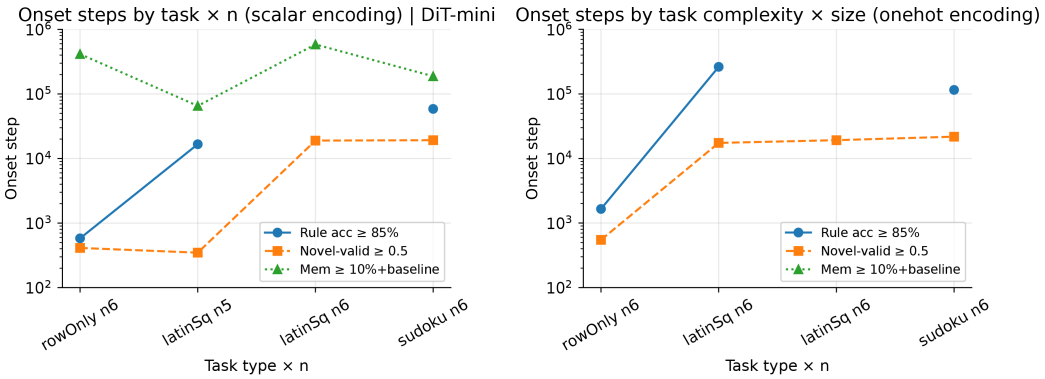


Figure 25: **Onset times for rule learning, novel valid generation, and memorization in structured categorical tasks.** We compare DiT-mini training dynamics across RowOnly, Latin Square, and Sudoku rule families. **Left:** scalar encoding. **Right:** one-hot encoding. For each task, we measure the first training step at which the model reaches rule accuracy $\geq 85\%$ (blue circles), novel-valid fraction ≥ 0.5 (orange squares), and memorization ratio at least 10% above the corresponding baseline (green triangles). Note we used a slightly different threshold for τ_{rule} since some task didn't reach 90% accuracy. Onset times are plotted on a log scale. Across tasks, rule learning and novel-valid generation occur earliest for RowOnly rule and are delayed for more nested constrained rules such as Latin Square and Sudoku. Memorization emerges substantially later than rule learning in the scalar setting, and τ_{mem} is comparable across rule complexity, but is earlier for $n = 5$ case where the board is smaller. In the one-hot setting memorization does not happen or reach the threshold within the plotted training horizon for most tasks. These results extend the two-clock picture beyond parity: structured categorical rules also exhibit separable timescales for rule acquisition, novel valid generation, and memorization, with the absolute onsets depending on task structure and encoding.

C.7 Extended results for more naturalistic images and language

We report the more naturalistic image and language experiments of Sec. 5.2, using the task definitions and evaluation protocol in App. B.4.3.

C.7.1 Image results

Parity slows with group size; exact-K varies less. Parity acquisition slows dramatically with G: For `Orb`, onset rises from 22,335 steps at $G{=}2$ to 426,000 at $G{=}3$ (Tab. 6). For `Flower`, $G{=}2$ reaches sample-validity onset at 30,391 steps; $G{=}3$ reaches the group-level threshold at 730,000 and ends at sample validity 0.483; $G{=}4$ remains near chance within $2{\times}10^5$ steps. Exact-K varies less dramatically across K, taking 8,170–13,275 steps on `Orb` and 130,701–180,000 on `Flower`. In `Flower`, the slower learning of exact-K relative to parity $G{=}2$ is consistent with a cost of combining evidence across patch tokens for irregular petals when counting globally, although the datasets also differ in dimension and rendering.

Patch dimensionality strongly delays rule learning. The `Block` ladder retains the latent task, token count, and object alignment while varying patch dimension (Tab. 7). It gives $\tau_{\text{rule}} \propto d^{0.66}$ over $d \in [1, 192]$ ($R^2{=}0.997$); the ±1 anchor agrees closely with the synthetic run (744 versus 724 steps). At $d{=}192$, flat `Block` and shaded `Orb` runs with nuisance have nearly identical onsets (26,170 versus 26,213; Tab. 9), supporting patch dimension as a major source of delay in this comparison. `Flower` fits give $\tau_{\text{rule}} \propto d^{1.05}$ at fixed $T{=}144$ and $\tau_{\text{rule}} \propto T^{0.79}$ at fixed $d{=}192$ (Fig. 26; Tab. 8). These small sweeps also change projection sizes or measured petal overlap and do not isolate a universal tokenization law. For $d > h$, rank-deficient input and output projections limit noise cancellation under EDM; large-patch runs slow or fail within budget, and rendering conditions become consequential.

The innovation window remains open within budget. For $d \leq 12$, `Block` memorization begins at 1.2–1.7$\times10^5$ steps; for $d \geq 48$, no onset is observed within the `Block` budgets of 2.5–5$\times10^5$ steps, or in the tested $d{=}1728$ `Orb` runs through 10^6 steps. The $d{=}48, 192$ `Block` runs reach high validity while training matches remain near the uniform-valid-support baseline. Normalized pixel nearest-neighbor distances likewise rise from 0.42 at $d{=}12$ to 0.97–0.99 at $d{=}48, 192$. Thus, their valid, novel generation window remains open through the observed budget.

Table 6: Rule-learning onset τ_{rule} (steps) across domains, DiT-mini, robust criterion. Synthetic parity references are at $N{=}2048$; synthetic exact-K references at $N{=}4096$. Cross-domain ratios describe observed delays across configurations and do not isolate rendering effects: latent dimension, tokenization, and training-set size are not all matched. Rows use sample validity ≥ 0.5, except `Flower` parity $G{=}3$ and $G{=}4$, which use group-level validity ≥ 0.75 (sample validity never reaches 0.5 at $G{=}3$). †: budget-limited.

rule	synthetic ±1 1 bit/token	`Orb` ($P{=}8$) 1 obj/token	`Flower` (96 px, $P{=}8$) $\sim$5 tok/obj	`Orb`/synth	`Flower`/synth
parity $G{=}2$	724	22,335	30,391	30.8×	42.0×
parity $G{=}3$	10,512	426,000	730,000	40.5×	69.4×
parity $G{=}4$	55,120	—	never ($> 2{\times}10^5$)†	—	$> 3.6\times$
exact-K, $K{=}4$	804	8,170	130,701	10.2×	162.6×
exact-K, $K{=}6$	1,716	13,275	160,000	7.7×	93.2×
exact-K, $K{=}12$	2,196	13,139	180,000	6.0×	82.0×

Table 7: Source data for Fig. 5B (`Block` rows). `Block` ladder: parity G=2, D=36, N=2048, T=36, DiT-mini; patch dimension varies, changing the input/output projections and pixel-space denoising target. τ_{mem} uses 2× the null $N/2^{18} = 0.0078$. †: $d > h$, rank-deficient patch embedding.

case	d	τ_{rule}	τ_{mem}	final mem / null	steps
synthetic ±1	1	724	145,056	112.8×	10^6
raw ±1 anchor	1	744	140,000	94.5×	5×10^5
`Block` P=1	3	1,806	120,000	78.2×	5×10^5
`Block` P=2	12	4,193	165,000	60.6×	5×10^5
`Block` P=4	48	9,462	never	0.8×	5×10^5
`Block` P=8	192	26,170	never	1.0×	5×10^5
`Block` P=11	363	60,000	never	0.8×	2.5×10^5
`Block` P=12	432†	60,000	never	—	2.5×10^5
`Block` P=16	768†	145,000	never	1.1×	2.5×10^5
`Block` P=24	1728†	never	never	0.9×	5×10^5
`Orb` P=24, shaded, clean	1728†	145,000	never	1.0×	10^6

Table 8: Source data for Fig. 26, using readability-gated sample validity. `Flower` scaling cases, parity G=2, N=2048, DiT-mini: d at fixed T=144 (image and patch scaled together), and T at fixed d=192. Tokens per petal are measured from the decoder's blob labels. Fits: $\tau_{\text{rule}} \propto d^{1.05}$ (R^2=0.918, $d \leq 363$) and $\tau_{\text{rule}} \propto T^{0.79}$ (R^2=0.999). †: $d > h$, rank-deficient.

run	T	d	tokens / petal	τ_{rule}
48 px, P=4	144	48	4.2	5,422
72 px, P=6	144	108	4.5	8,804
96 px, P=8	144	192	4.8	30,391
132 px, P=11	144	363	5.1	38,004
192 px, P=16	144	768†	5.0	never ($> 2.5\times10^5$)
48 px, P=8	36	192	2.0	9,808
128 px, P=8	256	192	7.0	46,117

Table 9: Rendering ablations at fixed patch dimension. Parity G=2, N=2048, one object per token. The shaded `Orb` with geometric nuisance has diameter 4.8 px; the 5.8 px variant has fixed geometry. †: $d > h$, rank-deficient.

content	nuisance	d	τ_{rule}
flat `Block`	none	192	26,170
3D `Orb`, diameter 4.8 px	photometric + geometric	192	26,213
3D `Orb`, diameter 5.8 px	photometric	192	22,335
flat `Block`	none	1728†	never ($> 5\times10^5$)
flat disc	none	1728†	185,000
flat disc	photometric	1728†	195,000
3D `Orb`	none	1728†	145,000
3D `Orb`	photometric + geometric	1728†	340,000

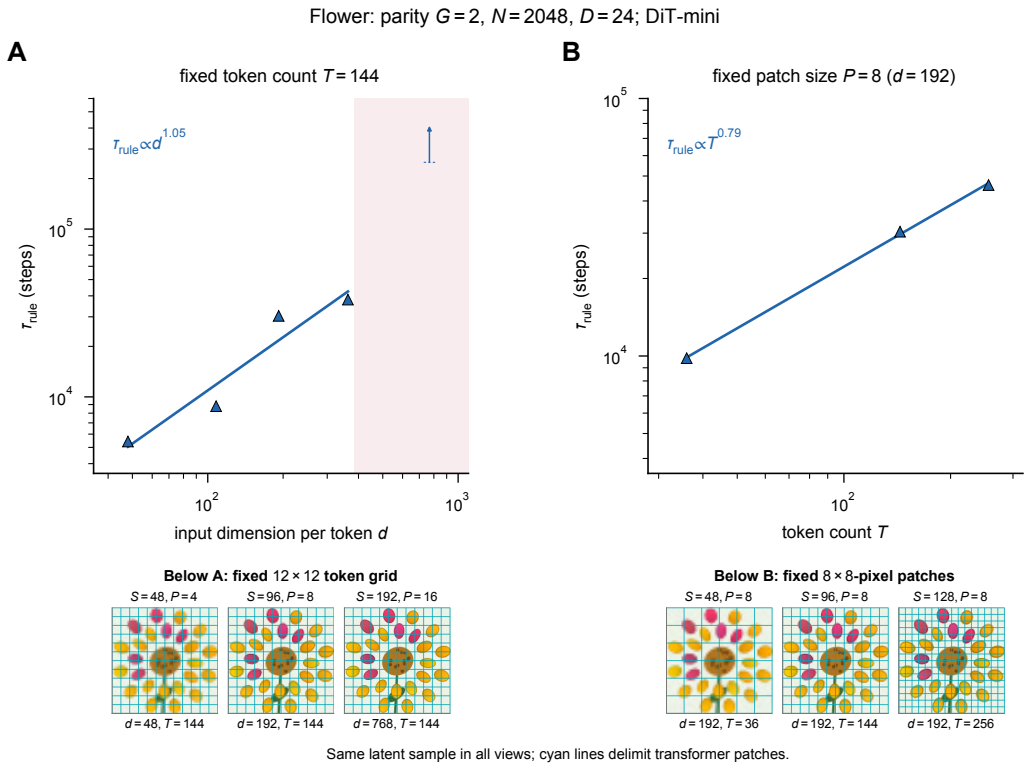


Figure 26: **Tokenization sweeps within `Flower`.** Both panels use $D{=}24$ latent bits, parity $G{=}2$, $N{=}2048$, and DiT-mini. **A.** Patch dimension $d = 3P^2$ varies at fixed $T{=}144$ by scaling image resolution and patch side length P together. **B.** Token count varies at fixed $d{=}192$ ($P{=}8$) by changing image resolution. Solid lines are log–log fits to observed onsets; the dashed boundary and arrow mark the 250,000-step budget without onset, excluded from the fit. Shading marks $d > 384$. **Below.** The same latent training example is shown at representative settings from **A** and **B**, with cyan lines delimiting $P{\times}P$ transformer patches: **A** retains a $12{\times}12$ token grid as image and patch size grow, whereas **B** uses progressively finer token grids at fixed $P{=}8$. Petal overlap varies slightly in **A** and more substantially in **B** (Tab. 8); these are within-dataset sweeps, not matched comparisons with `Block`.

C.7.2 Language results

Zero-shot generations often fail to terminate. Under zero-shot chat-template evaluation, Qwen3 yields no valid answers in 18 of 20 selected rule–prompt cells; narrative prompts predominantly produce overlong periodic outputs.

Fine-tuning opens a transient innovation window. Early checkpoints yield validity 0.995–1.000, training matches 0–0.010, and 1019–1024 distinct valid answers per 1024 draws, with narrative-prompt transfer validity 0.69–1.00 (Tab. 11). Memorization follows at 26–50 epochs, and all sampled d0 answers are training examples by epoch 500. This demonstrates a transient interval of valid, novel generation, followed by output concentration on the training set.

The language innovation window narrows with rule difficulty. Compared with GPT-mini from scratch at the same D, N, Qwen3 crosses the validity threshold 3.6–87 times earlier (Tab. 10). Validity-crossing time spans 44-fold across G for Qwen3 versus 128-fold for GPT-mini, whereas memorization onset varies less (26–50 versus 276–345 epochs). For GPT-mini at $G{=}6, 9$, $\tau_{\text{mem}} < \tau_{\text{rule}}$ orders crossings of different thresholds: $3/1024$ for training matches versus 0.5 for validity. At the later validity crossing, 81% and 87% of valid outputs are training examples, respectively; these runs show no distinct phase of high-validity, predominantly novel generation. Qwen3 exhibits such a phase directly (Tab. 11), with $\tau_{\text{mem}}/\tau_{\text{rule}}$ narrowing to 1.18 at $G{=}9$. Model scale, architecture, training, and sampling also differ, so the comparison does not isolate pretraining's causal contribution.

Table 10: Source data for Fig. 5C. `Light-switch` parity, $D{=}36$, $N{=}4096$: validity and training-match threshold crossings in epochs for Qwen3-1.7B SFT and GPT-mini from scratch, median-smoothed persistence criterion. Qwen3 memorization is conditional on valid answers; GPT memorization is unconditional. The fixed $3/1024$ threshold floor is used for both, although GPT evaluation has 2048 draws. Ratios are $\tau_{\text{mem}}/\tau_{\text{rule}}$; the two onsets use different thresholds, so their ordering alone does not establish generalization. †: validity crosses 0.5 with outputs dominated by training examples.

G	support	Qwen3 τ_{rule}	Qwen3 τ_{mem}	ratio	GPT-mini τ_{rule}	GPT-mini τ_{mem}	ratio
2	$2.6{\times}10^5$	1.0	30.7	30.7×	3.6	284.2	78.9×
3	$1.7{\times}10^7$	2.2	26.1	11.9×	18.8	276.2	14.7×
4	$1.3{\times}10^8$	1.6	36.2	22.6×	105.2	344.5	3.3×
6	$1.1{\times}10^9$	5.0	36.2	7.2×	432.5†	328.5	0.76×
9	$4.3{\times}10^9$	42.7	50.2	1.18×	464.5†	344.5	0.74×

Table 11: The "innovative" state shortly after τ_{rule} (Qwen3-1.7B SFT, 1024 samples): nearly every draw is valid and novel, and validity partially transfers to the three narrative prompts (d1–d3: 52, 99, 177 words) never seen in training: better for $G = 2, 4, 6$, less well for $G = 3, 9$. The innovative state is different from the end of training (at 500 epochs), validity is still 1.000 but the memorized fraction is 1.000.

G	epoch	valid	memorized	novel-valid	distinct valid / 1024	transfer d1 / d2 / d3
2	2.2	1.000	0.010	0.990	1023	0.977 / 0.996 / 1.000
3	3.1	1.000	0.000	1.000	1024	0.898 / 0.766 / 0.805
4	2.2	1.000	0.000	1.000	1024	0.934 / 0.953 / 0.926
6	7.0	1.000	0.000	1.000	1024	0.984 / 0.984 / 0.953
9	42.7	0.995	0.000	0.995	1019	0.797 / 0.691 / 0.770

C.8 Optimization ablations

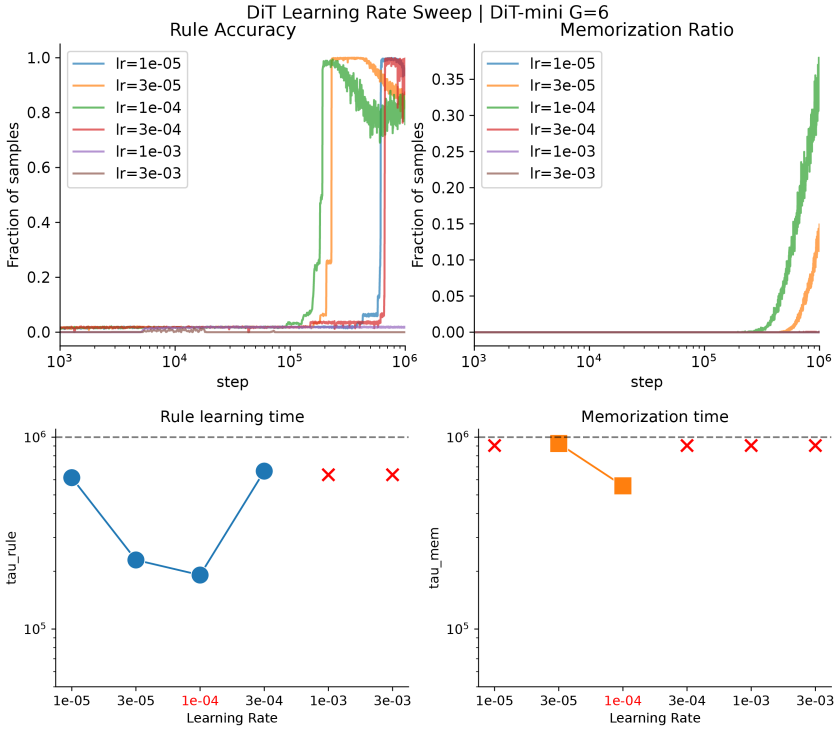


Figure 27: **Learning-rate sweep for DiT-mini on group parity with $G = 6$.** We train DiT-mini models with learning rates ranging from 10^{-5} to $3{\times}10^{-3}$ and track rule accuracy, memorization ratio, rule-learning time τ_{rule}, and memorization time τ_{mem}. Moderate learning rates enable rule learning, with $\text{lr} = 10^{-4}$ producing the earliest reliable rule-learning transition, followed by $\text{lr} = 3{\times}10^{-5}$ and 10^{-5}. Larger learning rates fail to acquire the rule within the training budget, while $\text{lr} = 3{\times}10^{-4}$ reaches high rule accuracy only very late. Memorization is observed primarily for the learning rates that successfully learn the rule, with $\text{lr} = 10^{-4}$ showing the strongest late memorization. Red crosses denote runs for which the corresponding onset time was not reached within the training horizon; gray dashed lines mark the training budget.

We systematically ablate the optimization parameters, i.e. learning rate and weight decay for both the DiT and GPT models, keeping all other hyperparameters at their respective baselines. Onset times for rule learning and memorization are detected using a *sustained threshold* criterion: the first checkpoint at which the metric exceeds the threshold for 5 consecutive evaluation points, to avoid counting transient spikes. The thresholds are: Sample Accuracy > 0.90 for rule learning, and Sample Memorization Ratio > 0.10 for memorization, rule onset time later than memorization time are discarded since these are usually due to memorization.

DiT learning rate sweep. We sweep $\eta \in \{10^{-5}, 3{\times}10^{-5}, 10^{-4}, 3{\times}10^{-4}, 10^{-3}, 3{\times}10^{-3}\}$ using the Adam optimizer (no weight decay), 10^6 gradient steps, N=4096, G=6, D=36 (Fig. 27). The baseline is η=10^{-4} (Adam, λ=0). Very low learning rates (10^{-5}) fail to reach the rule-learning transition within 10^6 steps; very high rates (10^{-3}–$3{\times}10^{-3}$) destabilize training. The intermediate range $\eta \in [3{\times}10^{-5}, 3{\times}10^{-4}]$ reliably produces both transitions. Interestingly, the value used in main experiment 10^{-4} yields the earliest rule learning onset and the memorization onset, showing our setting is well suited for learning the task. Though tuning up or down the learning rate can indeed delay the memorization onset.

DiT weight decay sweep. We sweep $\lambda \in \{10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 5{\times}10^{-2}\}$ using AdamW at fixed η=10^{-4}, 10^6 steps, all other settings identical to the DiT baseline (Fig. 28). The DiT baseline uses λ=0 (Adam), so this sweep explores whether L_2 regularization slows or suppresses memorization. We didn't find strong effect of weight decay on τ_{rule} or τ_{mem} in the range we explored.

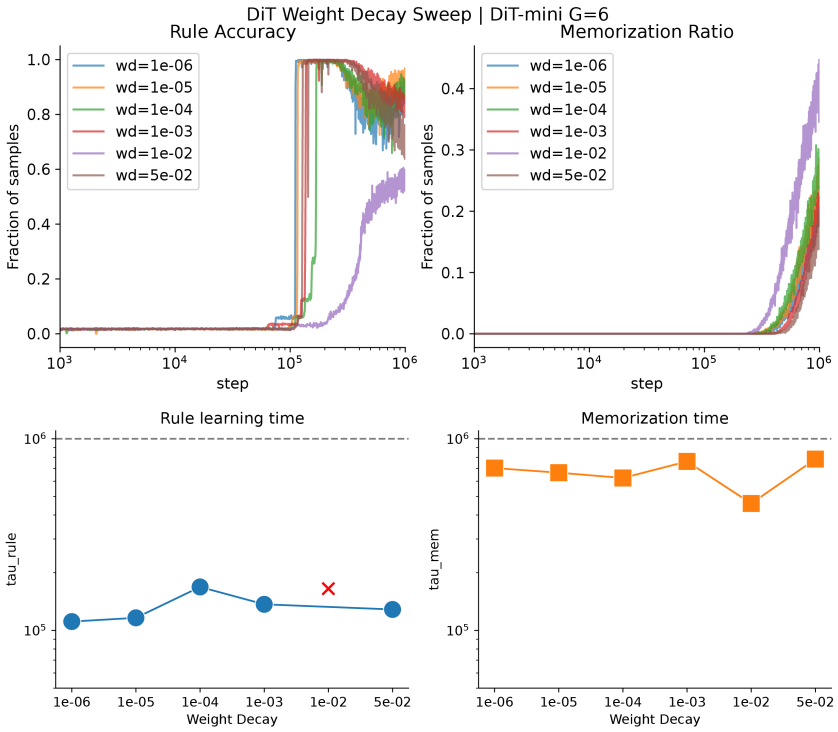


Figure 28: **Weight-decay sweep for DiT-mini on group parity with $G = 6$.** We train DiT-mini models across a range of weight decay values while keeping the learning rate fixed. Rule learning occurs robustly across small to moderate weight decay values, with τ_{rule} remaining on the order of 10^5 steps. Very large weight decay delays and weakens rule acquisition, as seen for $\text{wd} = 10^{-2}$, which does not reach the rule-learning criterion within the training budget. In contrast, memorization onset τ_{mem} is comparatively insensitive to weight decay over most of the tested range, remaining near late training for successful runs. These results indicate that DiT rule learning at $G = 6$ is more sensitive to learning rate than to moderate weight decay. Red crosses denote missing onset times; gray dashed lines mark the training budget.

GPT learning rate sweep. We sweep $\eta \in \{10^{-5}, 3{\times}10^{-5}, 10^{-4}, 3{\times}10^{-4}, 10^{-3}, 3{\times}10^{-3}\}$ using AdamW with fixed weight decay λ=0.01, batch size 256, 10^5 gradient steps, N=4096, G=6 (Fig. 29). We found that the baseline setting 10^{-4} is also the fastest to reach the rule transition, and the memorization time. Increasing learning rate shortens both τ_{rule} and τ_{mem} until a certain point 10^{-3} where it won't learn or memorize.

GPT weight decay sweep. We sweep $\lambda \in \{0, 10^{-3}, 10^{-2}, 5{\times}10^{-2}, 3{\times}10^{-1}, 1, 10\}$ using AdamW at fixed η=10^{-4}, all other settings identical to the GPT-mini baseline (Fig. 30). We find generally increasing the weight decay delays the memorization time τ_{mem} although rule learning is also affected by it. We find that a moderate weight decay of λ=$3{\times}10^{-1}$ maximizes the temporal gap between rule-learning onset and memorization onset, consistent with the grokking literature [Power et al., 2022], where weight decay suppresses memorization more than it delays rule generalization. At very large λ ($\lambda \geq 1$) the effective learning signal is attenuated and both transitions slow, reducing the observable separation.

D Broader Impacts

This work is a foundational study of rule learning and memorization in controlled generative models, and does not introduce deployed systems, human-subject data, or high-risk datasets. Its positive impact is to provide diagnostics for when generative models learn abstract constraints versus merely fit finite training sets, which may help improve reliability in structured generation, scientific modeling, and reasoning systems. The main negative risk is indirect: methods that improve rule-conforming generation could eventually strengthen generative systems in domains where misuse is possible, and

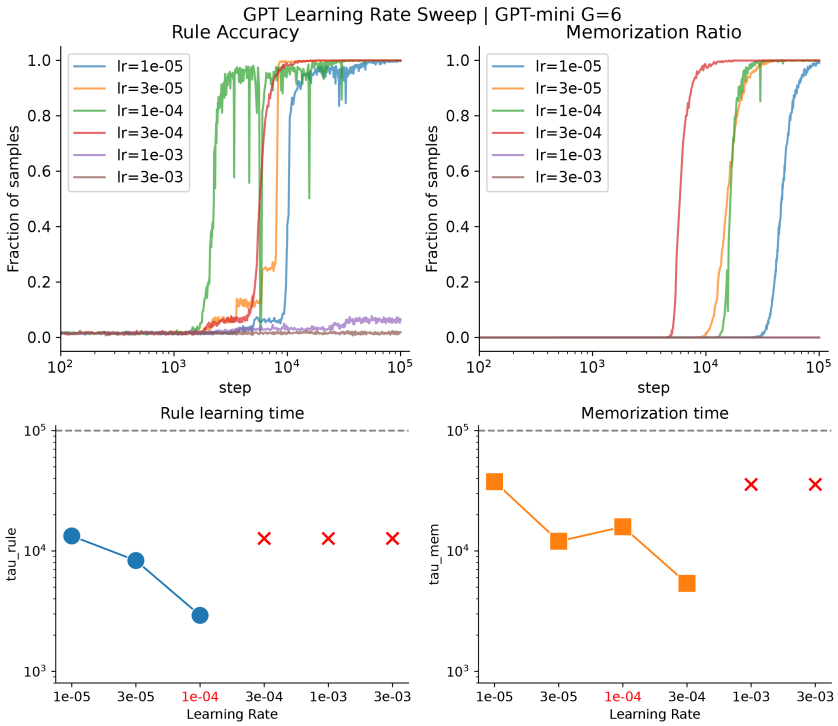


Figure 29: **Learning-rate sweep for GPT-mini on group parity with $G = 6$.** We train GPT-mini models with learning rates from 10^{-5} to 3×10^{-3} and measure rule accuracy, memorization ratio, τ_{rule}, and τ_{mem}. Rule learning is strongly learning-rate dependent: intermediate learning rates, especially $\text{lr} = 10^{-4}$ and 3×10^{-4}, reach high rule accuracy rapidly, whereas larger learning rates fail to learn the rule. Memorization follows successful rule learning and occurs substantially earlier in GPT than in DiT, reaching high memorization ratios for the learning rates that learn the task. This sweep shows that GPT has a narrower stable learning-rate window, but within that window both rule learning and memorization proceed rapidly. Red crosses denote runs that did not reach the corresponding onset criterion within the training budget; gray dashed lines mark the training budget.

overinterpreting transient rule-learning behavior could lead to misplaced trust in models that later memorize or violate constraints. We mitigate these risks by focusing on synthetic tasks, reporting failure modes alongside successes, and emphasizing evaluation over deployment.

E LLM Usage

Experimental use. We use `Qwen3-1.7B` [Qwen Team, 2025] as an experimental model for the `Light-switch` task, evaluating zero-shot generation, supervised fine-tuning, and transfer across prompt formulations. The model source, license, prompts, training protocol, and evaluation settings are described in App. B.4.3.

Research and writing assistance. Generative AI tools, including large language models, were used during manuscript preparation and research workflow support. Their use included visualizing results for submission, implementing standard methods, editing text for grammar and word choice, data processing and filtering, facilitating or running experiments, and drafting sections of the paper. All scientific claims, experimental choices, analyses, and final manuscript content were reviewed and controlled by the authors.

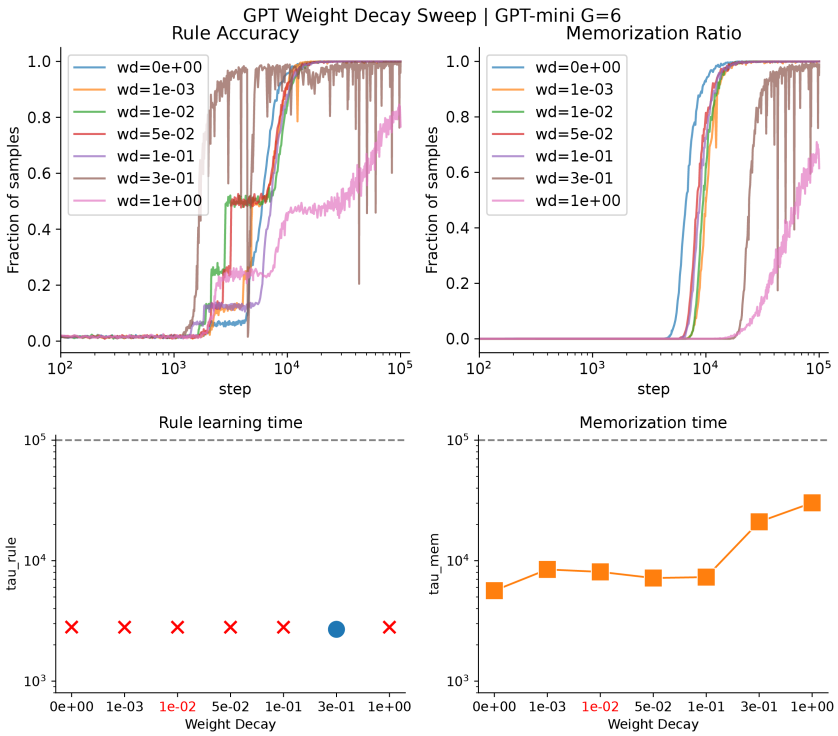


Figure 30: **Weight-decay sweep for GPT-mini on group parity with $G = 6$.** We train GPT-mini models across weight decay values from 0 to 1 and track rule accuracy, memorization ratio, τ_{rule}, and τ_{mem}. Across most weight decay values, GPT-mini rapidly reaches high rule accuracy, indicating that rule learning is robust to moderate regularization. However, memorization time is strongly affected by large weight decay: increasing weight decay delays τ_{mem}, with $\text{wd} = 0.3$ and $\text{wd} = 1$ producing substantially later memorization. Strong regularization can therefore slow memorization without necessarily preventing rule acquisition, although very large weight decay also destabilizes or weakens the rule-accuracy trajectories. Red crosses denote onset times that were not reached within the measured window; gray dashed lines mark the training budget.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction state the two-timescale framing, the innovation-window claim, the architecture comparison, and the score-landscape dissection; these claims are supported by the main empirical sections and the extended analyses in the appendix.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The discussion frames the work as a controlled synthetic-rule study and discusses scope limits, including high-interaction rules, implications for natural data, and directions needed before making broader deployment claims.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [N/A]

Justification: The paper does not present formal theorems or proofs. The energy, score, and attractor-basin analyses are used as explanatory and diagnostic tools rather than as formal theoretical results.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: The experiments use synthetic data-generating rules, and the appendix specifies the architectures, hyperparameters, sampling procedures, evaluation metrics, and task-generation protocols needed to reproduce the main reported results.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [No]

   Justification: The official project repository at https://github.com/Animadversio/GenRuleLearningDynamics links to the existing research implementation, including procedural dataset generators and training and analysis scripts. The implementation README describes the code structure and main entry points, and the appendix specifies experimental settings; a complete reproduction environment and exact commands for all reported results are not yet provided.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Training, model, sampling, and evaluation details are presented in §B.1 and §B.3; task definitions and GPT details are given in §B.4 and §B.5.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

Justification: Due to the large training cost, multiple runs of all configurations are not feasible. However, we indeed run multiple training repeats for key training setups and critical parameters e.g. all G for $N = 4096$, DiT-mini, esp. the $G = 6$ case as the rule learning is not stable in these boundary cases. Thus, errorbar for some configurations can be seen in Fig. 2. Further, when scaling laws are estimated, we denoted the goodness of fit and n that go into the fit. For sample level analysis, we have much larger n and statistical analysis are provided.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Appendix B.2 reports hardware, per-run runtimes, run counts, and GPU-hours, with separate accounting for synthetic models, rendered-image runs (including partial runs), and Qwen3 SFT. It also states which additional research costs and upstream pretraining are excluded.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research uses synthetic datasets and controlled small-model experiments, does not involve human subjects or sensitive data, and to the best of our knowledge conforms to the NeurIPS Code of Ethics.

Guidelines:

- The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: Appendix D discusses potential positive impacts for reliable structured generation and scientific modeling, as well as indirect misuse and misplaced-trust risks.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The work does not release high-risk pretrained models, scraped datasets, or deployed generative systems; the experiments use synthetic rule-generated data and controlled models.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.

- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: Appendix B.4.3 identifies and credits the Qwen3 checkpoint and tokenizer (Apache-2.0), the adapted DiT code (CC BY-NC 4.0), and the adapted EDM code (CC BY-NC-SA 4.0), with links to their sources and licenses. These assets are used for noncommercial academic research; the task datasets are procedurally generated.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The public code repository includes a README documenting its organization, dataset generators, and training and analysis entry points. The appendix documents the procedural datasets and experimental settings.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The work does not involve crowdsourcing, human-subject experiments, or participant data.

    Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The work does not involve crowdsourcing, human-subject experiments, or participant data, so IRB approval or equivalent review is not applicable.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [Yes]

    Justification: Appendices B.4.3 and E describe Qwen3-1.7B as an experimental model for zero-shot generation and supervised fine-tuning, including its source, license, prompts, and evaluation protocol. Appendix E also discloses LLM assistance with research workflows and manuscript preparation; the authors reviewed and controlled all scientific claims and final content.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.